Interactivité en Id pour le projet Monsoon

Sylvain HUET. X91

Avril-Juin 1994

Contents

Préambule

Le projet Monsoon est mené au MIT par l'équipe du professeur Arvind, en collaboration avec Motorola. Il vise l'étude approfondie, théorique et expérimentale, de l'architecture dataflow parallèle.

L'équipe du MIT a dans cette optique conçu une machine dataflow, appelée Monsoon, dont on peut trouver maintenant quelques exemplaires dans différentes universités américaines. Elle a également développé un langage de programmation adapté à ce nouveau type d'architecture — le langage Id, fonctionnel, parallèle et non-strict — , ainsi que les outils nécessaires à son utilisation — compilateurs divers —, dont la réalisation a mis en évidence de nouveaux problèmes théoriques.

Pour Motorola, ce projet est une voie de travail intéressante dans le domaine des machines parallèles. L'esprit qui l'anime est sans doute bien résumé dans une de ses plaquettes de présentation :

"Imagine a computer program that can run on thousands of processors with no modifications and even greater performance. Scalability and programmability would be virtually unlimited. The consequences of such capabilities are destined to take the computer world by storm."

Jusqu'à présent Monsoon a surtout été utilisé pour étudier le parallélisme de certains algorithmes, les entrées-sorties ne servant guère qu'au début et à la fin des applications, pour lire les données et écrire le résultat. J'ai donc proposé de développer une bibliothèque permettant d'utiliser les principales routines d'X-Window directement à partir d'une application écrite en Id. Cette bibliothèque recouvre plusieurs domaines d'X-Window et notamment la gestion des événements, ce qui ouvre la voie à la programmation d'applications interactives.

Chapter 1

Environnement général

1.1 Description de l'architecture dataflow

1.1.1 Introduction

L'objectif de l'architecture dataflow est de rompre avec le séquentialisme de Von Neumann. L'idée principale est de considérer un programme comme un graphe dont les noeuds sont des opérations simples, et le long duquel les données se déplacent (on appelle ces 'paquets de données' des 'tokens').

Par exemple, l'addition est représentée par un noeud à deux entrées et une sortie. Lorsque deux tokens arrivent au noeud (un sur chacune des deux entrées), l'addition est calculée et un token contenant le résultat est émis sur la sortie. Note importante : le premier token attend le second, et c'est au moment ou le second arrive que l'addition est effectuée.

On peut de la même manière modéliser tous les opérateurs, y compris les opérateurs booléens, ce qui permet de réaliser très facilement des boucles. On peut ensuite généraliser à toutes sortes de noeuds, p entrées, q sorties, sachant que les q sorties sont calculées au moment où un p-ième token atteint la dernière entrée libre.

Le fonctionnement d'une machine dataflow peut donc être schématisé ainsi :

- le programme est représenté par la liste des noeuds et leur interconnexion
- les données (ou *tokens*) en déplacement sont stockées dans une queue
- une unité de calcul pioche dans la queue, avance le token jusqu'au noeud suivant. Si le token achève de remplir les entrées du noeud, l'opération correspondant au noeud est effectuée, et le ou les tokens résultats sont

stockés dans la queue. Sinon, le token est simplement attaché au noeud, en attendant les tokens suivants.

Cette description est évidemment très simplifiée car elle ne rend pas compte de la possibilité d'exécuter plusieurs calculs indépendants dans une même partie du graphe au même moment. Cependant elle donne une idée générale de l'architecture dataflow et notamment des possibilités de parallélisation : il suffit d'avoir plusieurs unités de calcul qui piochent dans la queue de tokens. Le point fondamental est que le programme est indépendant de la parallélisation.

1.1.2 Gestion des tokens : ETS (Explicit Token Store)

Le modèle ETS, utilisé pour Monsoon décrit exactement, et d'une manière très proche du hardware final, le fonctionnement d'un processeur dataflow. Il se limite cependant à des graphes dont les noeuds ont au plus deux entrées.

Plusieurs questions se posent assez rapidement :

- lorsqu'un token arrive sur un noeud, comment déterminer rapidement si un autre token est déjà arrivé sur l'autre entrée ?
- comment gérer les sous-procedures dans les graphes dataflow ?
- comment permettre plusieurs calculs simultanés dans une même partie du graphe ?
- où stocker toutes les informations temporaires ?

Le modèle ETS répond à ces questions, moyennant un travail supplémentaire pour le compilateur. En effet, le programme ne sera pas fourni sous la forme d'un seul graphe, mais sous celle de plusieurs graphes (chacun étant appelé un *bloc*).

Chaque bloc peut être considéré comme une sous-fonction : c'est un graphe dataflow dont les entrées numérotées seront remplies par les arguments passés lors de l'appel du bloc, et la sortie sera un noeud qui retourne le résultat au bloc appelant. Cependant le bloc est aussi construit de manière telle qu'il n'est pas possible d'avoir à un moment donné, pour un appel donné du bloc, plusieurs jeux de tokens arrivant à un même noeud : les boucles dans le graphe d'un bloc ne sont possibles que dans la mesure où il y a une séquentialisation implicite. Cependant il est permis d'avoir plusieurs exécutions simultanées d'un même bloc avec des données différentes. C'est au compilateur de découper le graphe initial en blocs répondant à cette condition.

A chaque exécution d'un bloc, une zone de mémoire est attribuée qui permet de stocker l'état de chaque noeud du bloc : pour chaque noeud, la place exacte

(*l'offset*) où sera stocké cet état est également définie par le compilateur : elle est inscrite dans le programme. L'état d'un noeud est soit :

- vide, si aucun token n'est arrivé sur l'une des entrées,
- plein, si un token est déjà arrivé, auquel cas ce token représente l'état du noeud.

Un token est composé de plusieurs champs :

- un marqueur (ou *tag*) contenant :
  - l'adresse IP du noeud vers lequel il est dirigé
  - l'adresse FP de la zone mémoire affectée pour l'exécution du bloc
- la valeur du token qui peut être :
  - un entier
  - un flottant
  - un pointeur
  - un tag

Le traitement d'un token est donc le suivant :

- lire l'état du noeud où le token arrive : pour cela on lit l'offset x associé à l'instruction (dont l'adresse IP est codée dans le token), et on lit l'état à l'adresse $(FP + x)$,
- si le noeud est vide, stocker le token à l'adresse $(FP + x)$,
- si le noeud contient déjà un token, effectuer l'opération, et créer le ou les tokens résultats. Vider l'adresse $(FP + x)$.

De par la condition imposée lors de la définition des blocs (voir plus haut), on est sûr que lorsqu'un token arrive sur un noeud contenant déjà un autre token, ces deux tokens sont effectivement "liés" : il ne peut pas y avoir simultanément deux calculs différents sur un même noeud.

1.1.3 Parallélisme

Le modèle de parallélisme présenté en introduction était volontairement très simplifié ; cependant il avait le mérite de montrer l'intérêt principal de l'architecture dataflow dans les ordinateurs parallèles : le programme ne dépend pas de la parallélisation, le programmeur n'a pas à se soucier du nombre de processeurs dont il dispose, il peut faire tourner son application sur n'importe quelle configuration, sans avoir à la modifier.

Répartition des tokens

On voit clairement dans le modèle ETS que certaines zones mémoire sont critiques (c'est-à-dire qu'il y est fait constamment référence) :

- la zone mémoire contenant la queue de tokens.
- la zone mémoire affectée à l'exécution d'un bloc,

L'idée retenue pour le projet Monsoon est de ne pas partager ces zones entre les différents processeurs. Il s'ensuit que :

- chaque processeur gère sa propre queue de tokens,
- l'exécution d'un bloc se fait entièrement sur un seul processeur.

Cela se réalise très simplement : la partie du token qui code l'adresse de la zone mémoire allouée pour l'exécution du bloc se divise en deux,

- le numéro du processeur sur lequel le bloc va être exécuté,
- l'adresse du bloc dans la mémoire de ce processeur.

La répartition des tâches entre les différents processeurs se fait donc au moment du lancement de l'exécution d'un bloc (et plus particulièrement de l'affectation d'une zone mémoire pour ce bloc).

A la sortie du processeur, les tokens produits sont dirigés en fonction du numéro de processeur qu'ils contiennent : ou bien ils sont stockés directement dans la queue du processeur, ou bien il transitent via un réseau vers le processeur dont ils contiennent le numéro.

Accès à la mémoire partagée

Il reste une troisième zone de mémoire qu'il n'est pas possible de ne pas partager : c'est celle qui contient toutes les données qui ne sont pas seulement des entiers simples ou des flottant, par exemple tous les tableaux.

Ces données sont stockées dans des unités spéciales qui ont la même particularité que les processeurs : on peut en juxtaposer autant que l'on veut dans une machine.

Pour ne pas ralentir les processeurs, les instructions qui permettent de lire ou d'écrire dans ces structures sont en fait formulées sous la forme de tokens. Ces tokens sont un peu plus gros que les autres et se décomposent en deux parties :

1. un token normal qui indique le numéro de l'unité de stockage et l'adresse de la lecture ou de l'écriture.

2. la valeur à écrire pour une écriture, le tag de retour pour une lecture.

Il est à noter que la lecture se fait en deux temps : envoi d'une requête, puis réception d'un token contenant la valeur lue. Entre les deux, le processeur continue de traîter les tokens présents dans sa queue.

1.2 Description de Monsoon

(Les chiffres correspondent à la version développée en 1991).

Monsoon est composé de plusieurs parties distinctes :

- les PE (pour *processing element*) qui sont les unités de calcul de Monsoon,
- les IS (pour *I-Structure*) qui gèrent la mémoire partagée,
- un réseau qui gère les communications entre ces modules (qui se présentent chacun sous la forme d'une carte de type 9Ux400mm) et dont la bande passante est 800Mbytes/sec,
- des modules d'entrée-sortie, qui permettent les communications avec l'extérieur (réseau ou mémoire de masse), et qui sont à base de microprocesseurs 68030 (puisque le projet est mené en collaboration avec Motorola).

La version de base comprend simplement un élément de chaque.

Il est à noter que Monsoon ne possède aucune mémoire cache : la mémoire dédiée à chaque processeur est suffisamment rapide pour réaliser une opération en 1 cycle, alors que la mémoire partagée est de toutes manières rendue lente par la communication des tokens de requête.

Processing element

Chaque PE contient plusieurs parties :

- une unité de traitement des tokens, avec un pipe-line sur huit niveaux, permettant de traîter un token par cycle,
- une zone mémoire pour stocker le programme : 256KWord(32-bits),
- une zone mémoire pour stocker les queues de tokens : 64KWord(144-bits),
- une zone mémoire pour l'exécution des blocs : 256KWord(72-bits).

La vitesse de traîtement est de 10 millions de tokens par seconde.

I-Structure memory module

Les IS ont une capacité de 4MWord(72-bits), et permettent de réaliser 4 millions d'opérations par seconde.

Entrées-sorties

Les entrées sorties sont le point faible de la machine : la liaison se fait à 16KBit par seconde.

1.3 Principes généraux du langage Id

Le langage Id est un langage spécialement conçu pour la programmation d'applications parallèles. Il se décompose en trois niveaux :

- le coeur d'Id est un langage purement fonctionnel, et donc totalement déterministe, qui a la particularité d'être non-strict : l'évaluation d'une fonction peut commencer avant que tous les arguments soient disponibles.
- les *I-Structure* sont des structures de données dans lesquelles on peut écrire une fois et une seule. De par cette unicité elles garantissent encore le déterminisme de l'exécution des programmes, mais elles introduissent la notion d'effet de bord, en même temps que la notion d'état : une *I-Structure* peut être vide ou pleine.
- les *M-Structure* sont comparables aux *I-Structure* : elles sont soit vides, soit pleines, mais contrairement aux *I-Structure*, une *M-Structure* pleine peut être vidée et on peut alors la remplir avec une nouvelle valeur. Cela introduit un indéterminisme lors de l'exécution : il est donc nécessaire d'introduire une possibilité de séquentialisation, sous la forme de *barrières*.

1.3.1 Langage fonctionnel

Le coeur d'Id est un langage fonctionnel, assez proche de Caml, qui permet de manipuler des entiers, des flottants, des caractères, des chaînes de caractères, ... Il permet aussi d'utiliser des listes, et des tableaux.

Quelques exemples simples :

Fonction $f(x) = x + 1$

```
def f x = x+1 ;
```

Suite de Fibonnacci

```
def fibo n n0 n1 = if (n==0) then n0
                   else if (n==1) then n1
                   else (fibo (n-1) n0 n1) + (fibo (n-2) n0 n1) ;
```

Inversion de liste

```
def rever nil    l = l
|   rever (x:xs) l = rever xs (x:l) ;
```

nil est la liste vide, et l'inversion de la liste *liste* s'obtient par :

```
liste_inversee = rever liste nil ;
```

Quicksort

Dans le même genre d'idée, on peut réaliser facilement un programme de tri d'une liste d'entiers :

```
def fusionl nil b = b
|   fusionl (a:as) nil = (a:as)
|   fusionl (a:as) (b:bs) =
 if (lt~int a b) then (a:(fusionl as (b:bs)))
                        else (b:(fusionl (a:as) bs));

def moitie nil = (nil,nil)
|   moitie (x:nil) = ((x:nil),nil)
|   moitie (x:y:l) = { (a,b) = (moitie l) ;
                      in
                       ((x:a),(y:b)) };

def quicks nil = nil
|   quicks (x:nil) = (x:nil)
|   quicks (x:y:k) = { (a,b) = (moitie (x:y:k)) ;
                      in
                       fusionl (quicks a) (quicks b) };
```

Principales structures

Comme les exemples précédents le montrent, un programme Id repose sur un nombre réduit de structures syntaxiques.

- Définition d'une fonction.

  ```
  def Fonction arg_1 arg_2 ... arg_n = Expression ;
  ```

- Conditions.

```
if (condition) then Expression_1 else Expression_2 ;
```

- Blocs.

```
{ Declaration_1 ;
  Declaration_2 ;
  ...
  Declaration_n ;
 in
  Resulat };
```

Il est à noté que les différentes déclarations sont effectuées en parallèle, ainsi que le résultat. Evidemment, une interdépendance de certaines de ces expressions crée une séquentialisation implicite. Par exemple, dans le bloc suivant y ne sera calculé qu'après x :

```
{ x = f 1 ;  % f est une fonction qui retourne un entier
  y = x + 1 ;
...
in
...};
```

Par contre dans l'exemple suivant, $z * 2$ pourra être calculé avant x :

```
{ x = f 1 ;
  z = f 2 ;
  y = x + 2*z ;
...
in
...};
```

Tout ceci est très proche (et pour cause) de l'architecture dataflow : les tokens se déplacent tous en même temps dans l'arbre, mais les noeuds parce qu'ils ont plusieurs entrées, produisent des synchronisations.

- Boucles.

```
{ while (condition) do
    ...
    next x = x + 1 ;
    ...
  finally Resultat };
```

Il est possible, en utilisant *next* de redéfinir la valeur d'une variable pour la boucle suivante. Ceci crée, en partie ou en totalité selon les cas, une séquentialisation implicite. Le *Resultat* est évalué une seule fois, à la suite de la dernière boucle, lorsque la condition est devenue fausse.

- *Pattern Matching*
  Il est possible de faire dépendre du type ou de la valeur d'une variable le résultat d'une expression ; c'est le traditionnel *case of* :

```
{ case e of
  pat1 = e1
 |...
 |patN = eN };
```

Par exemple, on peut réécrir la fonction de Fibonnacci :

```
def fibo n n0 n1 =
{ case n of
     0 = n0
 |   1 = n1
 |.. i = (fibo (i-1) n0 n1) + (fibo (i-2) n0 n1)};
```

Le |.. permet de traiter les cas par défaut. On peut également tester le type d'une variable ; voici un exemple d'évaluation d'un arbre ou chaque noeud est une addition, et chaque feuille un entier :

```
type noeud = val i
           | add noeud noeud ;

def evalarbre node =
   { case node of
      (val n) = n
    | (add a b) = (evalarbre a)+(evalarbre b) };
```

Ce qui s'écrit encore :

```
type noeud = val i
           | add noeud noeud ;

def evalarbre (val n) = n
|   evalarbre (add a b) = (evalarbre a)+(evalarbre b) ;
```

1.3.2 Les tableaux

Id permet de définir trois types de tableaux : les tableaux purement fonctionnels, les *I-Structure* et les *M-Structure*.

Les tableaux purement fonctionnels

Ce type de tableau n'utilise pas de notion d'état : lors de sa création, toutes les cases du tableaux sont évaluées. Par exemple le tableau des n premiers carrés s'obtient par :

```
tableau = { array (1,n) of [i] = (i*i) || i <- 1 to n } ;
```

On peut diversifier les sources d'initialisation. Par exemple avec $m < n$, et deux fonctions entières f et g :

```
tableau = { array (1,n) of
      | [i] = (f i) || i <- 1 to m
      | [i] = (g i) || i <- (m+1) to n } ;
```

Les *I-Structure*

Les *I-Structure* introduisent une notion d'état : chaque case peut être vide ou pleine. Lorsque le tableau est créé, toutes les cases sont vides, et il est alors possible de remplir chacune de ces cases une fois. Une seconde tentative d'écriture provoque une erreur. Par contre une tentative de lecture dans une case vide suspend cette lecture jusqu'à ce que la case soit remplie.

La présence des *I-Structure* permet de simplifier certains calculs. Soit par exemple le problème suivant : on dispose d'un tableau A d'entiers positifs et négatifs, et on veut retourner un tableau B qui contient les mêmes entiers que A, mais où les entiers positifs ont tous des indices supérieurs à ceux des entiers négatifs. Ce problème, assez incommode avec des tableaux fonctionnels, se résoud facilement avec une *I-Structure* : il suffit de lire le tableau A séquentiellement et de remplir B avec les valeurs lues, par la gauche lorsqu'elles sont négatives, par la droite lorsqu'elles sont positives.

```
def classe_signe A =
{ B = I_array (1,n) ;
  l = 1 ; % pointeur gauche, pour les nombres negatifs
  r = n ; % pointeur droit, pour les nombres positifs
  _ = { for j <- 1 to n do
        val = A[j] ;
        (next l, next r, k) = if (val<0) then ((l+1), r, l)
                                         else (l, (r+1), r);
        B[k] = val } ;
in
  B };
```

Les *M-Structure*

Les *M-Structure* sont un troisième type de tableau : tout comme les *I-Structure*, elles sont dotées de la notion d'état : une case peut être vide ou pleine. Les principales caractéristiques des *M-Structure* sont :

- lire le contenu d'une case la vide, et permet donc une écriture ultérieure,
- lire une case vide attend que la case soit remplie,
- écrire dans une case déjà pleine provoque une erreur.
- pour bien montrer la différence avec les autres tableaux, une syntaxe particulière a été adoptée. Lire le i^{eme} élément du tableau T et mettre le résultat dans x s'écrit :

  ```
  ...          % T[i] est soit vide, soit plein
  x = T![i] ;  % le contenu de T[i] est lu (attente si T[i] est vide)
  ...          % a present, T[i] est vide
  ```

  De même, stocker la valeur de x dans la même case s'écrit :

  ```
  ...          % on suppose que T[i] est vide
  T![i] = x ;  % on place la valeur x dans T[i]
  ...          % a present, T[i] est plein
  ```

- il est possible de manière atomique de lire une case et de réécrire imméditatement la même valeur (ce qui permet de lire une donnée du tableau sans l'altérer puisqu'une lecture simple vide la structure)

  ```
  ...          % T[i] est soit vide, soit plein
  x = T!![i] ; % le contenu de T[i] est lu (attente si T[i] est vide)
  ...          % T[i] contient encore x apres la lecture
  ```

De même, il est possible de manière atomique de vider une case et d'y placer immédiatement une nouvelle valeur :

```
...          % T[i] est soit vide, soit plein
T!![i] = x ; % on lit la valeur de T[i] (qui est alors perdue)
             % et on la remplace par la valeur x.
...          % a present, T[i] est plein
```

Les *M-Structure* permettent de résoudre certains problèmes peu commodes à traiter avec des tableaux fonctionnels. Par exemple, soit A un tableau contenant des entiers compris entre 0 et $nmax$, on veut créer le tableau B tel que pour tout $0 \leq i \leq nmax$, $B[i]$ est le nombre d'occurences de i dans A. Les *M-Structure* permettent de régler ce problème en temps linéaire $O(nmax + taille(A))$.

```
def classe_signe A =
{ B = { M_array (0,nmax) of [i] = 0 || i <- 0 to nmax } ;
      % initialialisation de B : toutes les cases a 0
  _ = { for j <- 1 to n do
        val = A[j] ;
        B![val] = B![val] + 1 } ;
in
  B };
```

L'introduction des *M-Structure* signifie également l'introduction de l'indéterminisme. En effet dans le programme suivant, il est impossible de prédire le résulat.

```
def random n =
{ A = {M_array (0,0) of 0 }; %initialisation d'une case M-structure
  i = 1 ;
  _ = { while (i<n) do
        next i = i + 1 ;
        A!![0] = i };
 in
  A!![0] };
```

De plus cet indéterminisme peut provoquer des erreurs : imaginons que l'on veut placer la valeur d'une *M-Structure* dans x, et qu'on veut ensuite y écrire la valeur y, sans rapport avec x. La solution suivante est incorrecte :

```
...
x = A![i] ;
A![i] = y ;
...
```

En effet, rien n'assure que la lecture aura lieu *avant* l'écriture, et si l'écriture est effectuée en premier, $A[i]$ étant déjà plein, il y aura une erreur (avec interruption de l'exécution).

Pour remédier à cela le concept de *barrière* a été introduit. L'exemple précédent s'écrit alors :

```
...
x = A![i] ;
---
A![i] = y ;
...
```

La barrière représentée par les trois tirets impose que les déclarations qui la précèdent soient toutes évaluées avant de passer à celles qui suivent. Evidemment un usage intensif des barrières diminue le parallélisme, mais, notamment dans l'utilisation des *M-Structure*, leur emploi est souvent indispensable.

Revenons une dernière fois sur l'exemple précédent : supposons que y dépende de x, par exemple $y = x + 1$. Cette dépendance crée une séquentialisation implicite, et la barrière est alors inutile :

```
...
x = A![i] ;
A![i] = x + 1 ;
...
```

1.3.3 Evaluation

Une question importante lors de l'utilisation d'effets de bord (et la plupart des fonctions graphiques sont des effets de bord) est de savoir comment fonctionne l'évaluation en Id : toutes les définitions seront-elles évaluées ? A quel moment est-on sûr qu'une définition a été évaluée ? Quand le résultat d'une fonction est-il retourné ?

Ordre d'évaluation

Comme on l'a déjà vu, aucun ordre d'évaluation n'est imposé : les définitions et le résultat d'un bloc sont évalués en parallèle. La séquentialisation ne peut se faire que de deux manières :

- **implicitement :** une définition dépend d'une autre,
- **explicitement :** par l'utilisation de barrières.

Ce qui est effectivement évalué

Soit l'exemple suivant :

```
{ y = f 1 ;
  x = 2 ;
in
  x };
```

Le résultat de ce bloc est x, mais comme on le voit, il est inutile de calculer y pour évaluer x. Cependant la fonction f pourrait contenir des effets de bord (écriture dans une *I-Structure* ou dans une *M-Structure*, fonction graphique). La question est donc de savoir si l'on considère que ces effets de bord doivent être considérés comme faisant partie du résultat de la fonction ou non. Les concepteurs d'Id ont choisi de répondre par l'affirmative : y sera donc évalué, bien qu'il ne soit d'aucune utilité pour le calcul du résultat du bloc.

Il y a cependant une exception notable : la structure $if...then...else...$. En effet, chacune des expressions A et B dans *if (cond) then A else B* peut contenir des effets de bord : par conséquent la condition *cond* est évaluée en premier, puis, selon le résultat, seule une des deux expressions A et B est alors évaluée.

Disponibilité du résultat

Dans l'exemple précédent, on a vu que le résultat x ne dépendait pas de la valeur de y : il est donc possible que la valeur de x (à savoir 2) soit disponible avant la fin du calcul de y. Dans ces conditions deux solutions sont possibles :

- attendre qu'il n'y ait plus de redex dans le bloc avant de libérer le résultat
- transmettre le résultat dès qu'il est disponible, mais continuer la réduction du bloc.

Cette fois c'est la seconde solution qui a été retenue. La première peut cependant s'obtenir en utilisant une barrière :

```
{ y = f 1 ;
  x = 2 ;
 ---
 in
  x };
```

Si l'on ne met pas la barrière, on est assuré que la valeur de y sera calculée, mais on ne sait pas à quel moment : cela s'effectuera en parallèle avec la suite du programme.

Chapter 2

Réalisation de l'interface X

2.1 Gestion des entrées-sorties en Id

Sur Monsoon, les entrées-sorties ne sont pas implémentées de manière très performante. En fait le processus se décompose en 8 étapes, réparties entre Monsoon (en Id) et la machine Unix (en C) :

1. (en Id) allocation d'une structure spéciale,
2. (en Id) remplissage de cette structure avec les arguments,
2. (en Id) interruption de Monsoon : toute la machine est arrêtée,
3. (en C) lecture des arguments de l'interruption,
4. (en C) traitement de l'interruption,
5. (en C) (éventuellement) écriture du résultat dans Monsoon,
6. (en C) relancement de Monsoon,
7. (en Id) (éventuellement) lecture du résultat,
8. (en Id) libération de la structure allouée en 1.

La structure utilisée pour gérer les I/O contient entre autres dix mots libres à l'usage de l'utilisateur, qui permettent de passer des arguments (chaque mot contient 64bits, ce qui permet de stocker indifférement entiers, flottants et pointeurs). Deux mots sont également prévus pour stocker le résultat. Si cela s'avère insuffisant, il est possible de réserver un buffer de taille quelconque : c'est particulièrement utile pour passer des chaînes de caractères, ou des tableaux.

Les fonctions permettant d'allouer ou de désallouer ces structures, d'y lire ou d'y écrire, ainsi que celles commandant l'interruption appartiennent à la bibliothèque de base développée par l'équipe du MIT.

Une première ébauche de bibliothèque graphique avait été développée par Alejundro Caro en 1991, qui permettait d'ouvrir une fenêtre et d'y faire quelques dessins simples. Elle n'incorporait pas notamment la gestion des couleurs ni celle des événements, et comportait quelques erreurs.

2.2 Fonctions implémentées

Les fonctions implémentées sont tout simplement les principales fonctions de la Xlib :

- fonctions de connexion à un serveur X, lecture des paramètres standards,
- gestion des fenêtres : création, déplacement,...
- graphismes de base : points, lignes, cercles, polygones,...
- gestion des couleurs : création de palettes, définition de couleurs, ...
- gestion des événements : réception et envoi,
- divers : pixmap, copie de zones, lecture de la position de la souris, ...

D'autres fonctions peu nombreuses mais très pratiques ont été ajoutées.

2.2.1 Fonctions de connection

```
display = XOpenDisplay display_name ;
```

permet de se connecter au serveur *display_name*, retourne le pointeur *display* qui sera utilisé à chaque appel de fonction.

```
_ = XCloseDisplay display ;
```

permet de se déconnecter du serveur *display*.

```
screen = XDefaultScreen display ;
```

retourne l'identificateur de l'écran par défaut du serveur *display*.

```
root = XDefaultRootWindow display ;
```

retourne l'identificateur de la racine par défaut du serveur *display*.

```
gc = XDefaultGC display drawable ;
```

retourne l'identificateur du contexte graphique (GC) par défaut du serveur *display* et de type adapté à *drawable*.

```
visual = XDefaultVisual display screen ;
```

retourne l'identificateur du *visual* par défaut (utilisé lors de la création d'une nouvelle palette).

```
depth = XDefaultDepth display screen ;
```

retourne la profondeur par défaut de la mémoire écran de l'écran *screen* (par exemple 8 pour un écran 256 couleurs, 1 pour un écran monochrome).

```
white = XWhitePixel display screen ;
black = XBlackPixel display screen ;
```

retournent la valeur des points blancs et noirs dans la palette par défaut de l'écran *screen*.

```
colormap = XDefaultColormap display screen ;
```

retourne l'identificateur de la palette par défaut de l'écran *screen*.

```
_ = XFlush display ;
```

vide le buffer des requêtes adressées au serveur *display*.

2.2.2 Gestion des fenêtres

```
window = XCreateSimpleWindow display parent x y width height border_width
           border background ;
```

définit une nouvelle fenêtre dans le serveur *display*, de fenêtre parente *parent*, placée en (x, y) de dimensions $(width, height)$, avec un bord de largeur *border_width*, une couleur forme *border*, et une couleur fond *background*, et retourne l'identificateur de cette fenêtre.

```
_ = XDestroyWindow display window ;
```

détruit la fenêtre *window*.

```
_ = XSetWindowBackground display window background ;
```

modifie la couleur de fond de la fenêtre *window*.

```
_ = XMapWindow display window ;
```

pose la fenêtre sur l'écran, et la fait donc apparaître.

```
_ = XUnMapWindow display window ;
```

enlève la fenêtre de l'écran.

```
_ = XMoveWindow display window x y ;
```

déplace la fenêtre *window* en (x, y).

```
_ = XResizeWindow display window width height ;
```

modifie les dimensions de la fenêtre *window* : les nouvelles sont $(width, height)$.

2.2.3 Graphismes de base

```
gc = XCreateGC display drawable ;
```

crée un *contexte graphique* compatible avec le type de *drawable*.

```
_ = XFreeGC display gc ;
```

détruit le contexte graphique *gc*.

```
_ = XClearArea display window x y width height exposures_c_bool ;
```

efface une zone de la fenêtre *window*, et génère éventuellement un événement *Expose*.

```
_ = XClearWindow display window ;
```

efface le contenu de la fenêtre *window*.

```
_ = XDrawPoint display drawable gc x y ;
```

affiche le point de coordonnées (x, y) dans *drawable* (qui peut être une fenêtre ou un *pixmap*), selon le contexte graphique *gc*.

```
_ = XDrawPoints display drawable gc tab mode ;
```

affiche un ensemble de points dans *drawable* ; les coordonnées sont dans le tableau **fonctionnel** *tab* de type *Array (I,I)*. Le mode indique le choix du système de coordonnées (0 : coordonnées absolues, 1 : coordonnées relatives au point précédent, absolues pour le premier).

```
_ = XDrawLine display drawable gc x1 y1 x2 y2 ;
```

trace la ligne $(x_1, y_1) - (x_2, y_2)$ dans *drawable*.

```
_ = XDrawLines display drawable gc tab mode ;
```

trace une ligne brisée joignant un ensemble de points dans *drawable* ; les coordonnées sont dans le tableau **fonctionnel** *tab* de type *Array (I,I)*. Le mode indique le choix du système de coordonnées (0 : coordonnées absolues, 1 : coordonnées relatives au point précédent, absolues pour le premier).

```
_ = XDrawRectangle display drawable gc x y width height ;
```

trace le rectangle $(x, y) - (x + width, y + height)$ dans *drawable*.

```
_ = XDrawArc display drawable gc x y width height angle1 angle2 ;
```

trace dans *drawable* un morceau de l'ellipse contenue dans le rectangle $(x, y) - (x + width, y + height)$, et délimité par les angles *angle1* et *angle2* (les angles sont spécifiés en 64^{eme} de degré).

```
_ = XFillRectangle display drawable gc x y width height ;
```

dessine un rectangle plein $(x, y) - (x + width, y + height)$ dans *drawable*.

```
_ = XFillPolygon display drawable gc tab mode ;
```

affiche un polygone plein, délimité par un ensemble de points dans *drawable* ; les coordonnées sont dans le tableau **fonctionnel** *tab* de type *Array (I,I)*. Le mode indique le choix du système de coordonnées (0 : coordonnées absolues, 1 : coordonnées relatives au point précédent, absolues pour le premier).

```
_ = XFillArc display drawable gc x y width height angle1 angle2 ;
```

dessine dans *drawable* un quartier de l'ellipse contenue dans le rectangle $(x, y) - (x + width, y + height)$, et délimité par les angles *angle1* et *angle2* (les angles sont spécifiés en 64^{eme} de degré).

```
_ = XDrawString display drawable gc x y string ;
```

affiche dans *drawable* la chaîne de caractères *string*. (x, y) sont les coordonnées du coin inférieur gauche. Le fond n'est pas dessiné : la chaîne de caractères se superpose sur le fond.

```
_ = XDrawImageString display drawable gc x y string ;
```

affiche dans *drawable* la chaîne de caractères *string*. (x, y) sont les coordonnées du coin inférieur gauche. Le fond est redessiné : un rectangle couleur fond est affiché avant la chaîne de caractères.

```
_ = XSetBackground display gc background ;
```

règle la couleur de fond du contexte graphique *gc*.

```
_ = XSetForeground display gc background ;
```

règle la couleur de forme du contexte graphique *gc*.

```
_ = XSetFunction display gc function ;
```

règle la fonction graphique associée au contexte graphique *gc*. Les différents codes possibles sont (on pourra trouver leur signification exacte dans une documentation X-Window) :

```
GXclear
GXand
GXandReverse
GXcopy
GXandInverted
GXnoop
GXxor
GXor
GXnor
GXequiv
GXinvert
GXorReverse
GXcopyInverted
GXorInverted
GXnand
GXset
```

```
_ = XSetLineAttributes display gc line_width line_style cap_style join_style ;
```

règle la fonction graphique associée au contexte graphique *gc*, lors d'un tracé de ligne. Les différents codes possibles sont (on pourra trouver leur signification exacte dans une documentation X-Window) :

```
pour line_style :
    LineSolid
    LineOnOffDash
    LineDoubleDash

pour cap_style :
    CapNotLast
    CapButt
    CapRound
    CapProjecting

pour join_style :
    JoinMiter
    JoinRound
    JoinBevel
```

```
font = XLoadQueryFont display string ;
```

demande au serveur de charger en mémoire la fonte dont le nom est *string*, et retourne un identificateur de cette fonte (qui vaut 0 en cas d'erreur).

```
_ = XFreeFont display font ;
```

libère la mémoire utilisée pour la fonte.

```
_ = XSetFont display gc font ;
```

définit la fonte à utiliser lors des fonctions *XDrawString* et *XDrawImageString*.

2.2.4 Gestion des couleurs

```
colormap = XCreateColormap display window visual ;
```

crée une nouvelle palette vide associée à l'écran qui contient la fenêtre *window*, et retourne l'identificateur de cette palette.

```
_ = XFreeColormap display colormap ;
```

détruit la palette *colormap*.

```
color = XAllocNamedColor display colormap string ;
```

retourne le code de la couleur *string*, éventuellement créée dans la palette *colormap* si elle n'existait pas encore.

```
color = XAllocColor display colormap red green blue ;
```

retourne le code de la couleur *(red, green, blue)* —chaque valeur étant un entier compris entre 0 et 65535—, éventuellement créée dans la palette *colormap* si elle n'existait pas encore.

```
_ = XSetWindowColormap display window colormap ;
```

affecte la palette *colormap* à la fenêtre *window* : le window manager effectuera lui-même la substitution de palettes au cours de l'utilisation de plusieurs fenêtres avec des palettes différentes.

2.2.5 Gestion des événements

```
_ = XSelectInput display window event_mask;
```

définit le masque de réception des événements pour la fenêtre *window*. Les masques disponibles sont :

```
NoEventMask
KeyPressMask
KeyReleaseMask
ButtonPressMask
ButtonReleaseMask
EnterWindowMask
LeaveWindowMask
ButtonMotionMask
ExposureMask
StructureNotifyMask

event = XNextEvent display ;
```

attend le prochain événement. Le résultat *event* est de le forme suivante :

```
XEvent =  XKeyPress        win x y x_root y_root state keycode ascii
        | XKeyRelease      win x y x_root y_root state keycode ascii
        | XButtonPress     win x y x_root y_root state button
        | XButtonRelease   win x y x_root y_root state button
        | XMotionNotify    win x y x_root y_root state
        | XEnterNotify     win x y x_root y_root state
        | XLeaveNotify     win x y x_root y_root state
        | XExpose          win x y width height
        | XMapNotify       win
        | XConfigureNotify win x y width height
        | XDefault         type
```

Le *type* de XDefault retourne le code de l'événement, qui donc ne fait pas partie de ceux implémentés ici, c'est-à-dire autre que :

```
KeyPress
KeyRelease
ButtonPress
ButtonRelease
MotionNotify
EnterNotify
LeaveNotify
Expose
MapNotify
ConfigureNotify
```

L'état *state* des boutons de la souris et des touches spéciales du clavier peut être interprété à l'aide des masques suivants :

```
ShiftMask
LockMask
```

```
ControlMask
Button1Mask
Button2Mask
Button3Mask
```

Dans le cas des événements clavier, un appel à XLookupString est automatiquement réalisé, et le résultat est retourné dans *ascii*.

```
event = XCheckTypedEvent display event_type ;
```

regarde si dans la liste des événements se trouve un événement du type *event_type*. Si oui, le retourne de la même manière que XNextEvent. Si non, retourne l'événement *(XDefault 0)*.

```
_ = XSendEvent display window event_mask event ;
```

envoie l'événement *event* à la fenêtre *window*. *event* est du type *XEvent* (voir plus haut).

```
(root,child, root_x, root_y,
   win_x, win_y, state) = XQueryPointer display win ;
```

retourne la position de la souris, ainsi que l'état des ses boutons et des touches spéciales du clavier.

2.2.6 Divers

```
pixmap = XCreatePixmap display window width height depth ;
```

crée un *pixmap* pour l'écran qui contient *window*, de taille $(width, height)$, et de profondeur *depth*. Retourne l'identificateur de ce nouveau pixmap que l'on utilisera comme n'importe quelle fenêtre lors des opérations graphiques usuelles.

```
_ = XFreePixmap display pixmap ;
```

détruit le pixmap.

```
_ = XCopyArea display src dest gc x y width height dest_x dest_y ;
```

effectue une copie de la zone $(x, y) - (x + width, y + height)$ du *src* vers la zone $(dest_x, dest_y) - (dest_x + width, dest_y + height)$ de *dest*. *src* et *dest* peuvent être aussi bien des fenêtres que des pixmaps (et même un de chaque), mais il doivent avoir la même profondeur.

```
_ = XSynchronize display onoff ;
```

si *onoff=True*, active le mode synchronisé : chaque appel X est immédiatement pris en compte ; il n'y a pas de queue de requêtes.

```
_ = WaitForMap display ;
```

attend le prochain événement *MapNotify*. En effet si, entre le moment où l'on effectue un XMapWindow et le moment où la fenêtre est effectivement affichée à l'écran, on appelle quelques fonctions graphiques dans cette fenêtre, celles-ci seront perdues. Il peut donc être utile d'attendre l'affichage effectif d'une fenêtre, à l'aide de *WaitForMap*.

```
_ = DisplayArray display drawable gc x_win y_win tab colors alpha
      beta width_pix height_pix ;
```

affiche le contenu d'une matrice dans *drawable*, en utilisant (et en en modifiant la couleur forme) le contexte graphique *gc*.

- la matrice à afficher est la matrice **entière** et **fonctionnelle** *tab*
- la liste des couleurs à utiliser est donnée par le vecteur **fonctionnel** *colors*, et la couleur c de chaque valeur v de la matrice est déterminée par :

$$c = colors \left[\frac{v+\beta}{\alpha} \right]$$

- la matrice sera affichée à partir du point (x_{win}, y_{win}) de *drawable*, sous la forme d'un quadrillage où chaque valeur de la matrice *tab* sera représentée par un rectangle de taille $(width_{pix}, height_{pix})$.

2.3 Réalisation pratique de l'interface

Monsoon n'est pas une machine multi-utilisateur : si la machine Unix qui le dirige permet via le réseau de commander Monsoon de n'importe quel terminal, il n'est pas possible de travailler à plusieurs en même temps. De plus, la liaison I/O très lente ne rend pas le développement très facile : une fois qu'un programme en Id est compilé, il faut encore le charger dans la machine. L'utilisation de Monsoon pour le développement n'est donc pas recommandée.

Un outil palie cette insuffisance : il existe un émulateur de Monsoon (appelé MINT) qui tourne sur les stations suns, et qui réagit exactement comme la vraie machine, avec deux différences de vitesse :

- Monsoon calcule beaucoup plus vite,
- Mint a des I/O plus rapides.

Le développement sur Mint est donc plus agréable (même si le temps de chargement est encore important), et ne monopolise aucune resource autre qu'un peu de temps CPU sur une Sun.

Mon projet ajoutait une autre contrainte : il ne se limitait pas au développement d'une application en Id, il modifiait également le fonctionnement de la machine Unix censée diriger Monsoon : chaque fonction X nécessite deux bouts de code, un en Id, l'autre en C. Cela signifie qu'à chaque modification de la partie écrite en C, il fallait recompiler le serveur, et relancer une nouvelle session (ce qui prend environ 5 minutes sur Mint, ou 30 minutes sur Monsoon). Comme toujours en informatique (depuis les cartes perforées jusqu'aux cassettes audio comme mémoire de masse principale), les contraintes ont une vertu pédagogique : elles incitent à programmer juste du premier coup.

Chapter 3

Exemples de réalisation

Pour démontrer les nouvelles possibilités offertes par la librairie graphique, j'ai été amené à développer un certain nombre d'applications. Les deux premières sont de ma propre initiative et ont été développées sur Mint, avant que les routines C ne soient compilées pour Monsoon. Leur présentation (notamment celle du premier ...) a convaincu le laboratoire qu'il était temps de tester ces routines sur la vraie machine (processus assez long qui nécessitait une excursion dans les locaux de Motorola pour récupérer des fichiers nécessaires à la compilation).

Les deux suivantes ne se contentent pas de démontrer les possibilités interactives, mais abordent des questions de parallélisme très intéressantes. Le tableur est une idée d'Alejundro Caro, alors que le problème des n corps est une suggestion du Professeur Arvind. Ces deux applications m'ont permis d'approcher de plus près l'architecture dataflow et ont donné une touche plus théorique à mon travail.

3.1 Breakout

Breakout est un programme qui montre les possibilités nouvelles que la librairie graphique offre à Monsoon. Il s'agit d'un casse-brique digne de 'pong' : l'affichage est monochrome dans une fenêtre à fond noir, la raquette et les briques sont représentées par des rectangles blancs, la balle par un carré. Le contrôle se fait avec les boutons de la souris : un pour déplacer la raquette vers la gauche, un pour la déplacer vers la droite, un autre pour quitter le programme.

Apres avoir initialisé quelques paramêtres standards, le programme définit un tableau qui contient l'état de chaque brique. C'est une *M-structure* qui sera utilisée dans la boucle du jeu pour détecter les chocs.

La boucle principale se décompose ainsi :

- lecture de la position et de l'état des boutons de la souris, par la fonction XQueryPointer : l'état des boutons n'est pris en compte que si le pointeur se trouve dans la fenêtre du casse-brique.
- déplacement éventuel de la raquette (deux fonctions XFillRectangle)
- test de collision de la balle avec les murs
- test de collision de la balle avec la raquette
- test de collision de la balle avec les briques (éventuellement une fonction XFillRectangle)
- déplacement de la balle (deux fonctions XFillRectangle)

Une astuce permet de n'imposer aucun ordre dans l'exécution des fonctions X-window au cours d'une boucle : tous les affichages de rectangles se font en appliquant la fonction logique EOR ('ou' exclusif) qui a la propriété d'être commutative. Il ne reste ainsi dans la boucle principale qu'une barrière, uniquement en cas de collision avec une brique, indispensable pour la gestion des *M-structures*, pour empêcher une tentative d'écriture dans une case déjà pleine.

Le programme donne une idée visuelle de la vitesse que l'on peut atteindre avec Monsoon, puisque le temps de calcul est pratiquement négligeable : il apparaît que cinquante à soixante appels par secondes sont le maximum que l'on peut espérer.

3.2 Jeu de la Vie

Ce programme est une simulation du jeu de la Vie qui met en évidence les possibilités de la nouvelle interface graphique. L'exécution se décompose en deux phases très différentes.

Définition de la position initiale

Au cours de cette phase, la fenêtre est couverte d'un quadrillage : chaque case est vide initialement, mais on peut y placer une cellule en cliquant dessus avec le bouton gauche de la souris. Recliquer dessus enlève la cellule.
Le bouton du milieu permet de réinitialiser tout le tableau, alors que le bouton droit permet de passer à la phase suivante.

Evolution

L'évolution est calculée en parallèle : à chaque pas, le programme compte le nombre de cellules présentes au voisinage de chaque case (le voisinage correspond aux 8 cases les plus proches), et détermine le nouvel état de la case.

- Une cellule survit si elle a deux ou trois voisins.
- Une cellule naît si elle a exactement deux voisins.

Le tableau de jeu est limité : une case d'un bord n'a que 5 voisins, et une case d'angle seulement 3, mais les règles d'évolution sont les mêmes.

Puis la nouvelle position est affichée. La librairie graphique permet de faire ceci en une seule interruption : en effet la fonction DisplayArray permet d'afficher dans une fenêtre quelconque le contenu d'un tableau à deux dimensions, en utilisant une palette quelconque, et en affichant pour chaque case de la matrice, des rectangles de dimension réglable.

Ici, le tableau à afficher est celui qui contient l'état de chaque case : 0 pour une case vide, 1 pour une case contenant une cellule. La palette est réduite à deux entrées : noir pour la valeur 0, blanc pour la valeur 1. Les points sont des carrés de 16x16.

Il est possible d'interrompre la simulation de deux manières différentes :

- Le bouton gauche stoppe l'évolution et revient à l'étape de définition.
- Le bouton droit sort du programme.

3.3 Excel

Cette application est un mini-tableur qui révèle une propriété très intéressante de l'architecture dataflow. Ce programme se limite au calcul sur les entiers, chaque cellule contient une expression qui fait intervenir d'autres cellules, des entiers, et les quatre opérations de base. L'évaluation de la feuille de calcul est effectuée dynamiquement : chaque fois que l'expression affectée à une cellule est modifiée, le programme détermine quelles cellules sont à recalculer, effectue ce calcul et affiche les résultats.

3.3.1 Grammaire

La grammaire utilisée est la suivante :

Expression : Expression + Produit
| Expression - Produit
| Produit

Produit : Produit * Terme
| Produit / Terme
| Terme

Terme : Entier
| Cellule
| (Expression)

Entier : - Séquence
| Séquence

Séquence : {0,...,9}
| {0,...,9} Séquence

Cellule : Alpha Séquence

Alpha : {A,...,Z}
| {A,...,Z} Alpha

Le programme effectue une analyse syntaxique qui à partir d'une chaîne de caractères retourne un arbre dont les noeuds sont l'une des quatres opérations et les feuilles soit un entier (éventuellement négatif), soit une cellule (sous la forme une séquence de lettres et une séquence de chiffres, par exemple : AB18).

3.3.2 Interface

Le programme gère une interface graphique qui permet à la fois d'éditer une expression, de l'affecter à une cellule, et de visualiser les résultats.

Dans la partie supérieure de la fenêtre se trouve une ligne d'édition dans laquelle, à l'aide du clavier (mais aussi, on le verra plus loin, de la souris), on peut construire une expression. Le reste de la fenêtre est un quadrillage où chaque case représente une cellule.

Chaque cellule contient deux lignes de cinq caractères :
— la premiere, en blanc sur noir reproduit l'expression attachée à la cellule (tronquée si elle occupe plus de cinq caractères),
— la seconde, en noir sur blanc, contient la valeur de la cellule, à savoir :

- rien si aucune expression n'est attachée à la cellule,

- "Undef" si la valeur ne peut être calculée (l'expression fait référence à une cellule vide ou également indéfinie, ou bien il y a une division par zéro),
- "Big" si la valeur ne peut être affichée sur cinq caractères,
- un entier dans les autres cas.

Une case 'Quit' est présente dans le coin de la fenêtre pour pouvoir quitter l'application.

L'interactivité se fait de la sorte :
— la construction de l'expression peut se faire exclusivement au clavier (touche *Backspace* disponible), mais, en cliquant sur une cellule avec le troisième bouton de la souris, on ajoute à la fin actuelle de l'expression les coordonnées de la cellule.
Typiquement pour faire la somme d'un certain nombre de cellules, on se contentera alternativement de cliquer sur la prochaine cellule, et d'appuyer sur la touche '+' du clavier.
— En cliquant sur une cellule avec le second bouton, on copie l'expression attachée à cette cellule dans la ligne de l'éditeur.
— En cliquant sur une cellule avec le premier bouton, on attache à cette cellule l'expression construite dans la ligne de l'éditeur. La feuille de calcul est mise à jour instantanément.

3.3.3 Description des structures de données utilisées

Quatre tableaux de type *M-structure* sont utilisés ; ils sont tous de la taille de la feuille de calcul. Pout toute cellule C, on a donc :

- **Expression :** le sommet de l'arbre représentant l'expression attachée à la cellule C, issu directement de l'analyseur syntaxique. Si aucune expression n'est attachée à C, cette case contient un noeud signalant qu'il n'y a pas d'expression : elle n'est pas vide au sens des *M-structure*.
- **Valeur :** la valeur de la cellule C, calculée à partir de l'expression précédente. Si aucune expression n'est attachée à C, cette case contient un code qui indique qu'il n'y a rien : elle n'est pas vide au sens des *M-structure*.
- **Liste des dépendances :** la liste des cellules dont l'expression fait référence à la cellule C. Eventuellement une liste vide. Cette liste est ordonnée pour accélérer les opérations telles que *recherche* ou *suppression*.
- **Drapeau :** utilisé lors de l'évaluation dynamique, pour éviter des recherches redondantes.

Mis-à-part le tableau des valeurs, les *M-structure* ne sont utilisées que comme mémoire modifiable : ces structures ne sont jamais dans *l'état vide*. Au contraire, le tableau des valeurs utilise pleinement cette possibilité de vider une case.

3.3.4 Mise à jour de la feuille de calcul : évaluation dynamique

Cette partie est de loin la plus intéressante du programme car elle met en évidence une propriété très particulière de l'architecture dataflow. Le problème à résoudre est le suivant :

- soit une feuille de calcul définie comme il est expliqué précédemment,
- on suppose qu'elle est à jour, à savoir que le tableau des valeurs correspond à celui des expressions,
- on remplace alors l'expression attachée à une cellule C, par l'expression E.
- on veut alors remettre à jour le tableau des valeurs, en un minimum de calculs, et en parallèle.

Commençons par un problème plus simple : on veut calculer la feuille complète. Pour cela on dispose d'un programme qui évalue une expression à partir de l'arbre issu de l'analyse syntaxique. Rappelons que les éléments de cet arbre sont de trois types :

- les quatre opérations
- une constante entière
- une référence à une autre cellule, c'est-à-dire une lecture du tableau des valeurs.

Il apparaît alors qu'une expression ne peut être calculée que lorsque les valeurs des cellules auxquelles elle fait référence sont disponibles. Si l'on veut programmer l'évaluation de la feuille complète sur une machine séquentielle, le principal problème sera donc de déterminer l'ordre dans lequel évaluer les cellules : l'évaluation d'une cellule ne peut commencer que lorsque toutes les cellules auxquelles elle fait référence ont été évaluées.

Avec l'architecture dataflow, tout est plus simple : en effet, une tentative de lecture dans une case vide est bloquante jusqu'à ce que cette case soit remplie. Il suffit donc de vider tout le tableau de valeurs, et de lancer *ensuite* l'évaluation de toutes les cellules *en parallèle*. Chaque fois qu'une expression est calculée,

son résultat est immédiatement stocké dans le tableau de valeurs. Ce faisant, le calcul des expressions qui dépendaient de cette valeur peut se poursuivre. En fait, le hardware dataflow trouve lui-même l'ordre d'évaluation des cellules !

Revenons au problème initial : on modifie l'expression attachée à la cellule C, et on veut mettre à jour la feuille de calcul. Ceci se fait en trois temps :

- détermination de la liste des cellules affectées par une modification de C,
- vérification que la nouvelle expression E ne crée pas de *boucle* dans la feuille de calcul,
- recalcul de la valeur des cellules de cette liste.

Liste des cellules affectées

On utilise dans cette phase le *tableau des dépendances*. En effet la liste des cellules dépendantes de C donne la liste des cellules directement modifiées par le changement d'expression de C. Mais si A dépend de C, les cellules dépendant directement de A (et donc présentes dans la liste de dépendance de A) vont être également modifiées : en parcourant récursivement cet arbre (puisque c'en est un), et en fusionnant les listes de dépendance, on obtient la liste recherchée.

Cependant, en procédant de la sorte, on prend le risque de parcourir plusieurs fois la même partie de l'arbre : une même cellule B peut se trouver à différents endroits dans la descendance de C. Prenons un exemple :

- A_0 et B_0 contiennent chacun une valeur quelconque
- pour $0 < i \leq N$, $A_i = B_i = A_{i-1} + B_{i-1}$

Chaque cellule a une liste de dépendance de longueur 2 : A_i et B_i ont tous deux pour liste de dépendance (A_{i+1} : B_{i+1} : nil). Si alors on modifie A_0 la cellule A_N (et sa descendance) sera visitée 2^{N-1} fois, ce qui fait exploser l'algorithme.

Pour remédier à ce problème, un autre tableau a été défini : il s'agit du *tableau des drapeaux*. Il est initialisé à 0.

A la i-ème évaluation d'une partie de la feuille de calcul, chaque cellule dont la liste de dépendances est lue se voit affecter la valeur i dans son drapeau. Et seules les listes de dépendances correspondant à une cellule dont le drapeau est différent de i sont examinées. Le jeu des *M-structure* permet d'éviter les problèmes de synchronisation : entre le moment où le drapeau est lu (et donc vidée), et le moment où la valeur i y est écrite, personne ne peut lire ce drapeau.

Le nombre de cellules visitées est alors exactement égal au nombre de branches du sous-arbre de sommet C. Pour avoir le temps de l'algorithme, il faut encore ajouter le temps de fusion des listes de dépendance. Dans tous les cas, l'algorithme est au pire en $O(n^2)$, où n est le nombre de cellules à recalculer.

Détection des boucles

Il y a boucle lorsque l'expression d'une cellule dépend implicitement de sa propre valeur : dans un tel cas, le calcul ne peut aboutir, et ne s'arrête pas. Il faut donc détecter les boucles lorsqu'on modifie une expression, avant de lancer le calcul.

Une autre manière de définir des boucles est la suivante : il y a boucle lorsque l'expression d'une cellule C fait référence à des cellules qui dépendent de la valeur de C. Ce qui se reformule ainsi : il y a boucle lorsque l'expression d'une cellule C fait référence à des cellules qui sont modifiées par un changement de l'expression affectée à C.

Autrement dit, on détectera les boucles en vérifiant qu'aucune des cellules auxquelles la nouvelle expression E fait *explicitement* référence n'est présente dans la liste des cellules affectées calculée à l'étape précédente.

Recalcul des valeurs

C'est une généralisation du calcul de la feuille complète : au lieu de vider tout le tableau des valeurs et de lancer le calcul de toutes les cellules, on ne vide et ne recalcule que les cellules présentes dans la liste des cellules affectées.

L'évaluation se fait de la même manière, en parallèle. Il est a noté que cette évaluation en parallèle n'est pas seulement un moyen d'accélérer le calcul sur une machine parallèle : il est *indispensable* que l'évaluation soit simultanée, sinon le blocage est généralement inévitable.

3.4 Problème des n-corps. Barnes-Hut

Le problème des n-corps consiste en la simulation des interactions gravitationnelles entre n astres. Typiquement ce problème se résoud en $O(n^2)$: on calcule chacune des interactions entre deux astres, et on effectue pour chaque astre la somme des forces qu'il subit (donc exactement $n(n-1)/2$ calculs d'interactions). La méthode de Barnes-Hut permet, moyennant un résultat approché, d'effectuer la simulation en $O(n \ln n)$.

3.4.1 Principe de l'algorithme

L'algorithme de Barnes-Hut consiste en une sectorisation de l'espace dont le résultat est un arbre ou chaque noeud possède 8 fils. Voici comment on procède pour créer cet arbre :

- on part d'un cube assez grand pour contenir tous les points (qui sera le sommet de l'arbre), puis on itère le processus suivant pour chaque point P.
- on positionne le point P dans la partition courante (à savoir l'une des feuilles de l'arbre courant) : il se trouve dans un cube C
- si le cube C est vide, on y place le point P
- si le cube C contient déjà un point Q, on découpe le cube C en huit, on place Q dans celui de ces huit cubes qui le contient, et on reprend l'algorithme pour le point P.

A la fin de l'algorithme, on obtient un arbre où chaque noeud représente un cube de l'espace (la taille d'un fils étant la moitié de celle du père), et où chaque feuille est soit vide, soit un astre.

Pour chaque noeud, on calcule le centre de gravité et la masse totale des astres présents dans sa descendance. On peut alors calculer l'accélération de chaque astre A par l'algorithme suivant :

- on commence par considérer le noeud N, sommet de l'arbre,
- on considère la condition C suivante : $d > s/\theta$, où d est la distance de A au centre de gravité de N, s la taille de la cellule (c'est-à-dire la longueur des arêtes), et θ un paramètre positif inférieur à $1/\sqrt{3}$ si l'on veut éviter tout problème,
- si C est remplie, on considère que les astres contenus dans N sont assez loin de A pour que leur action soit assimilée à un astre de même masse placé au centre de gravité,
- sinon, on calcule séparément, récursivement, l'action de chacun des huit fils de N sur A.

On peut donc espérer un calcul de seulement $n(7\log_8 n)/2$ interactions si l'arbre est bien équilibré.

3.4.2 Parallélisation du problème

Comme on le voit le problème se décompose en plusieurs parties indépendantes :

1. recherche d'un cube contenant tous les astres

2. construction de l'arbre, et calcul des centres de gravité
3. calcul de l'accélération
4. intégration de la vitesse et de la position

Recherche d'un cube contenant tous les astres

Il est possible de calculer les limites $(x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max})$ de n points en $O(\ln n)$ en parallèle :

- diviser l'ensemble E des points en deux sous-ensemble E_1 et E_2 de tailles égales (à une unité près)
- calculer récursivement les limites de chacun des deux sous-ensembles
- fusionner les deux résultats en utilisant les fonctions max et min
- si E est un singleton $\{A\}$, les limites de E sont égales aux coordonnées de A.

Une fois qu'on a calculé ces limites, on peut prendre le cube de sommet $(x_{min}, y_{min}, z_{min})$ et d'arête $max(x_{max} - x_{min}, y_{max} - y_{min}, z_{max} - z_{min})$.

Construction de l'arbre

La première idée est d'ajouter en parallèle les n astres dans l'arbre. Pour chaque astre A :

- lire le premier noeud
- si ce noeud est vide, y placer l'astre A.
- si ce noeud est un astre B, créer une subdivision du noeud, y placer B, et continuer l'algorithme pour A.
- si ce noeud est déjà une subdivision de l'espace reprendre l'algorithme avec le fils contenant A.

Cependant, pour éviter les conflits, c'est-à-dire pour traiter le cas où l'on essaye en parallèle de placer deux astres dans le même noeud, il faut effectuer une synchronisation : entre le moment où un astre *lit* un noeud, et le moment où il passe au suivant, personne ne doit avoir accès à ce noeud. Ceci peut se faire très simplement en utilisant les *M-Structure* : chaque noeud étant une *M-Structure*, l'algorithme s'écrit alors :

- lire le premier noeud (et donc vider la *M-Structure*)
- si ce noeud est vide, y placer l'astre A (et donc remplir la *M-Structure*).

- si ce noeud est un astre B, créer une subdivision du noeud, y placer B, placer cette nouvelle subdivision dans la *M-Structure* et continuer l'algorithme pour A.
- si ce noeud est déjà une subdivision de l'espace reprendre l'algorithme avec le fils contenant A (et remettre en place le contenu de la *M-Structure*)

Puisque la lecture d'une *M-Structure* vide provoque une attente, il n'y a plus de conflit possible. Cependant cela séquentialise l'accès aux structures partagées que sont les *M-Structure*. En fait de parallélisation, on obtient plutôt un pipe-line, puisque le traîtement de chaque astre commence par la lecture séquentialisée du premier noeud.

Une autre approche peut être envisagée, beaucoup plus parallèle, et totalement récursive :

- En entrée on dispose d'un ensemble de points E et d'un cube C les contenant tous,
- si E est vide, on retourne l'arbre nul,
- si E est un singleton, on retourne l'arbre réduit à un astre,
- sinon, on divise l'ensemble E en deux, en coupant le cube C selon $x = x_{milieu}$, où x_{milieu} est la coordonnée x du centre de C,
- on divise chacun de ces deux ensembles selon $y = y_{milieu}$,
- puis on divise encore selon $z = z_{milieu}$,
- on obtient alors huit ensembles (un par subdivision de C), et on recommence récursivement, en parallèle, pour chacun de ces ensembles, avec le sous-cube correspondant.

Il est également facile d'inclure le calcul du centre de gravité dans la construction de l'arbre : il suffit que la fonction récursive retourne non seulement l'arbre construit à partir de E, mais aussi le centre de gravité et la masse associés.

Notons que la parallélisation est presque totale puisque :

- les huits appels de la récursivité se font en parallèle, sans conflit d'aucune sorte,
- à chaque étape on peut avoir à couper 1 ensemble selon x, 2 selon y et 4 selon z, or les 2 selon y ainsi que les 4 selon z peuvent s'effectuer en parallèle,
- comme on va le voir maintenant, couper une liste en deux (selon x par exemple) peut être parallélisé.

En effet, soit une ensemble E de n points et $P : x = x_{milieu}$ un plan. Si les points de E sont rangés dans un tableau, il est possible de couper l'ensemble E selon le plan P en $O(\ln n)$.

Le fait d'utiliser un tableau (et non une liste) permet de couper en 2 l'ensemble de points en temps constant (et non linéaire dans le cas des listes). L'algorithme récursif et parallèle est le suivant :

- si E est vide, retourner $([], [])$,
- si E est un singleton $\{A\}$, lire la valeur de x_A ; si $x_A > x_{milieu}$, retourner $([A], [])$, sinon retourner $([], [A])$,
- sinon, couper E en deux sous-ensembles de tailles égales (à une unité près) F et G. lancer récursivement et en parallèle le calcul sur F et G, ce qui donne les résultats (F_{grand}, F_{petit}) et (G_{grand}, G_{petit}). On retourne alors $(F_{grand} \bigcup G_{grand}, F_{petit} \bigcup G_{petit})$. Si le nombre de processeurs est infini, cela se fait en temps constant.

Cet algorithme n'est intéressant qu'avec beaucoup de processeurs, puisqu'il est en fait en $O(n(\ln n)/p)$ avec comme limite $O(\ln n)$ lorsque p devient infini. Ceci est à comparer avec le $O(n)$ de l'algorithme séquentiel.

Calcul de l'accélération

Deux approches sont ici également possibles ; la première reste très proche de l'algorithme de Barnes-Hut tel qu'il est habituellement expliqué :

- pour chaque astre A, on effectue l'algorithme suivant en commençant par considérer le sommet N de l'arbre,
- si A est suffisamment loin du centre de gravité de N, calculer l'accélération correspondant à un seul astre placé en ce centre de gravité
- sinon, calculer récursivement l'accélération produite par chacun des huit fils de N

La parallélisation s'obtient en lançant le calcul de tous les astres en même temps. Cependant, comme pour l'étape précédente, l'arbre étant une structure partagée, le résultat sera plus proche du pipe-line que d'une vraie parallélisation du problème.

On peut cependant résoudre le problème avec une méthode comparable à celle développée à l'étape précédente :

- créer une *M-Structure* pour stocker le vecteur accélération de chaque astre, l'initialiser avec le vecteur nul, puis considérer l'ensemble E de tous les astres et le sommet N de l'arbre,

- couper l'ensemble E en deux :
  - F : ensemble des points situés assez près du centre de gravité de N,
  - G : ensemble des points situés loin du centre de gravité de N.

  Rappelons que le critère de sélection est : $d > s/\theta$. La division de E selon ce critère s'effectue de la même manière que la division par rapport à un plan précédemment expliquée : en parallèle, et en $O(n(\ln n)/p)$ avec comme limite $O(\ln n)$ lorsque p devient infini,
- pour les points de G, calculer l'accélération produite par un seul astre placé au centre de gravité de N (ceci se fait en parallèle),
- pour les points de F, calculer récursivement l'accélération produite par les huit fils de N (ces huit calculs s'effectuant en parallèle).

Intégration de la vitesse et de la position

Cette partie est de loin la plus facile : l'intégration se fait tout simplement par :
$x(t + dt) = x(t) + v(t).dt$
$v(t + dt) = v(t) + a(t).dt$

La parallélisation est évidente.

Remarque

Faute de temps, le test de l'algorithme n'a été réalisé que sur la version mono-processeur de Monsoon, et a été victime de son "efficacité" : comme on l'a vu dans la description du modèle ETS, l'exécution d'un bloc requiert l'affectation d'une zone mémoire qui permet de stocker l'état de chaque noeud ; or, en cas de parallélisation massive, le nombre de blocs à exécuter simultanément devient très important, ce qui en soit est plutot positif puisque cela indique qu'il sera possible de répartir la charge sur tous les processeurs disponibles.

Cependant, lors du test, cela a conduit à un dépassement de la capacité mémoire —et donc à un arrêt— pour seulement une cinquantaine d'astres. Deux solutions sont possibles :

- ne pas tout paralléliser.
- augmenter le nombre de processeurs,

3.4.3 Interface

L'interface développée pour cette application est une interface 3D qui permet de visualiser la position des astres et de changer de manière interactive la position

de la caméra.

La position de l'observateur est définie par trois valeurs ϕ, θ et ρ :

- la caméra pointe toujours vers le centre de la galaxie : le point de coordonnées $(0, 0, 0)$,
- ϕ représente la longitude,
- θ représente la lattitude
- ρ représente la distance de l'observateur au centre de la galaxie
- il n'est pas possible de faire une rotation autour de la direction de vision : la direction verticale de l'observateur reste tangente à la longitude.

La transformation permettant de passer des coordonnées absolues d'un point aux coordonnées dans le repère observateur a alors la forme suivante :

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -\sin\phi & 0 & \cos\phi \\ -\sin\theta.\cos\phi & \cos\theta & -\sin\theta.\sin\phi \\ -\cos\theta.\cos\phi & -\sin\theta & -\cos\theta.\sin\phi \end{bmatrix} . \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \rho \end{bmatrix}$$

Le contrôle se fait à la souris :

- déplacer la souris avec le bouton gauche enfoncé permet de modifier ϕ et θ,
- déplacer la souris avec le bouton du milieu enfoncé permet de modifier ϕ et ρ,
- le troisième bouton permet de quitter l'application.

La fonction *XDrawPoints* permet d'afficher tous les points en un seul appel X-Window.

Appendix A

Interface X, côté C : listing

```
/*****************************************************************************
 ***                                                                       ***
 ***                          Monsoon I/O Support                          ***
 ***                                                                       ***
 ***     Designed and implemented by members of the Computation            ***
 ***     Structures Group of the MIT Laboratory for Computer Science       ***
 ***                                                                       ***
 ***      Copyright (C) 1991  Massachusetts Institute of Technology        ***
 ***                            ALL RIGHTS RESERVED                        ***
 ***                                                                       ***
 ***         No copy of this source code may be made by any means,         ***
 ***         electronic or otherwise, without prior permission of the      ***
 ***         Massachusetts Institute of Technology.                        ***
 ***                                                                       ***
 *****************************************************************************/

/* General Comments:
 *
 * The image routines are not finished. Also, colormap manipulation
 * routines need to be added.
 */

#include "iographics.h"

/*** Monsoon System includes */
#include "mmi.h"
#include "iocommon.h"
#include "iosysdep.h"
```

```
/*** Other includes */
#include "stdlib.h"
#include "X11/Xlib.h"
#include "X11/Xutil.h"
#ifdef GNUC68k
#include "extraXlib.h"
#endif
/*************************************************************************
 *  DECLARATIONS, PROTOTYPES, and DEFINITIONS
 *  Commented portions are exported in iographics.h.
 *************************************************************************/

/* VOID io_initialize_graphics_io (VOID);
 * VOID io_reset_graphics_io (VOID);
 * VOID io_dispatch_graphics_op (VOID);
 * VOID io_terminate_graphics(VOID);
 */

VOID io_XOpenDisplay(VOID);
VOID io_XBlackPixel(VOID);
VOID io_XWhitePixel(VOID);
VOID io_XDefaultGC(VOID);
VOID io_XDefaultRootWindow(VOID);
VOID io_XDefaultScreen(VOID);
VOID io_XCloseDisplay(VOID);
VOID io_XDefaultVisual(VOID);
VOID io_XFlush(VOID);
VOID io_XCreateGC(VOID);
VOID io_XDefaultColormap(VOID);
VOID io_XDefaultDepth(VOID);
VOID io_XFreeGC(VOID);

VOID io_XCreateSimpleWindow(VOID);
VOID io_XDestroyWindow(VOID);
VOID io_XMapWindow(VOID);
VOID io_XSetWindowBackground(VOID);
VOID io_XUnmapWindow(VOID);
VOID io_XMoveWindow(VOID);
VOID io_XResizeWindow(VOID);

VOID io_XClearArea(VOID);
VOID io_XClearWindow(VOID);
VOID io_XDrawPoint(VOID);
```

```
VOID io_XDrawPoints(VOID);
VOID io_XDrawLine(VOID);
VOID io_XDrawLines(VOID);
VOID io_XDrawRectangle(VOID);
VOID io_XDrawArc(VOID);
VOID io_XFillRectangle(VOID);
VOID io_XFillPolygon(VOID);
VOID io_XFillArc(VOID);
VOID io_XDrawString(VOID);
VOID io_XDrawImageString(VOID);
VOID io_XSetBackground(VOID);
VOID io_XSetForeground(VOID);
VOID io_XSetFunction(VOID);
VOID io_XSetLineAttributes(VOID);

VOID io_XSelectInput(VOID);
VOID io_XNextEvent(VOID);
VOID io_XCheckTypedEvent(VOID);
VOID io_XQueryPointer(VOID);
VOID io_XSendEvent(VOID);

VOID io_XAllocNamedColor(VOID);
VOID io_XAllocColor(VOID);
VOID io_XCreateColormap(VOID);
VOID io_XSetWindowColormap(VOID);
VOID io_XFreeColormap(VOID);
VOID io_XLoadQueryFont(VOID);
VOID io_XFreeFont(VOID);
VOID io_XSetFont(VOID);

VOID io_XCreatePixmap(VOID);
VOID io_XFreePixmap(VOID);
VOID io_XCopyArea(VOID);

VOID io_WaitForMap(VOID);
VOID io_DisplayArray(VOID);

VOID io_XSynchronize(VOID);


/*** Error Function */
VOID io_invalid_graphics_op(auINT32 op);

/*** List utility functions and defs*/
```

```
typedef struct io_cons_cell {
  VOID *car;
  struct io_cons_cell *cdr;
} io_cons_cell;

io_cons_cell *io_cons(VOID *, io_cons_cell *);
VOID io_map_list (io_cons_cell *list, VOID (*fn)(VOID *arg));



#define IO_STD_OH 4

/* Operation Constants */

#define IO_XOPENDISPLAY_CODE 100
#define IO_XBLACKPIXEL_CODE 101
#define IO_XWHITEPIXEL_CODE 102
#define IO_XDEFAULTGC_CODE 103
#define IO_XDEFAULTROOTWINDOW_CODE 104
#define IO_XDEFAULTSCREEN_CODE 105
#define IO_XCLOSEDISPLAY_CODE 106
#define IO_XDEFAULTVISUAL_CODE 107
#define IO_XFLUSH_CODE 108
#define IO_XCREATEGC_CODE 109
#define IO_XDEFAULTCOLORMAP_CODE 110
#define IO_XDEFAULTDEPTH_CODE 111
#define IO_XFREEGC_CODE 112

#define IO_XCREATESIMPLEWINDOW_CODE 200
#define IO_XDESTROYWINDOW_CODE 201
#define IO_XMAPWINDOW_CODE 202
#define IO_XSETWINDOWBACKGROUND_CODE 203
#define IO_XUNMAPWINDOW_CODE 204
#define IO_XMOVEWINDOW_CODE 205
#define IO_XRESIZEWINDOW_CODE 206

#define IO_XCLEARAREA_CODE 300
#define IO_XCLEARWINDOW_CODE 301
#define IO_XDRAWPOINT_CODE 302
#define IO_XDRAWPOINTS_CODE 303
#define IO_XDRAWLINE_CODE 304
#define IO_XDRAWLINES_CODE 305
#define IO_XDRAWRECTANGLE_CODE 307
#define IO_XDRAWARC_CODE 309
```

```
#define IO_XFILLRECTANGLE_CODE 311
#define IO_XFILLPOLYGON_CODE 313
#define IO_XFILLARC_CODE 314
#define IO_XDRAWSTRING_CODE 316
#define IO_XDRAWIMAGESTRING_CODE 317
#define IO_XSETBACKGROUND_CODE 318
#define IO_XSETFOREGROUND_CODE 319
#define IO_XSETFUNCTION_CODE 320
#define IO_XSETLINEATTRIBUTES_CODE 321

#define IO_XSELECTINPUT_CODE 400
#define IO_XNEXTEVENT_CODE 401
#define IO_XCHECKTYPEDEVENT_CODE 402
#define IO_XQUERYPOINTER_CODE 403
#define IO_XSENDEVENT_CODE 404

#define IO_XALLOCNAMEDCOLOR_CODE 500
#define IO_XALLOCCOLOR_CODE 501
#define IO_XCREATECOLORMAP_CODE 502
#define IO_XSETWINDOWCOLORMAP_CODE 503
#define IO_XFREECOLORMAP_CODE 504
#define IO_XLOADQUERYFONT_CODE 505
#define IO_XFREEFONT_CODE 506
#define IO_XSETFONT_CODE 507


#define IO_XCREATEPIXMAP_CODE 600
#define IO_XFREEPIXMAP_CODE 601
#define IO_XCOPYAREA_CODE 602

#define IO_WAITFORMAP_CODE 900
#define IO_DISPLAYARRAY_CODE 901

#define IO_XSYNCHRONIZE_CODE 1000


/*** Copyright notice */
cSTRING io_graphics_copyright_notice =
"Copyright (C) 1991 Massachusetts Institute of Technology";

cSTRING io_DISPLAY_env_var = "DISPLAY";
io_cons_cell *IO_Displays = NULL;
```

```
/************************************************************************
 * FUNCTIONS
 ************************************************************************/

/* io_initialize_graphics_io - initialize graphics module.
 */
VOID io_initialize_graphics_io (VOID)
{
  return;
}

/* io_reset_graphics_io - resets the graphics module. */
VOID io_reset_graphics_io (VOID)
{
  return;
}

/* io_dispatch_graphics_op - dispatches to the appropriate X routine.
 * At the present, we require that at least 20 control words (actually 11
 * is the required MINIMUM for operation) are read by the io_interrupt_handler.
 * Otherwise, the X routines will not be able to read all their parameters.
 */
VOID io_dispatch_graphics_op (VOID)
{

  auINT32 graphics_op = io_host_ioreq_ptr[IO_OPCODE].ls;

  switch (graphics_op) {

  case IO_XOPENDISPLAY_CODE:
    io_request_type("XOpenDisplay");
    io_XOpenDisplay();
    break;

  case IO_XBLACKPIXEL_CODE:
    io_request_type("XBlackPixel");
    io_XBlackPixel();
    break;

  case IO_XWHITEPIXEL_CODE:
    io_request_type("XWhitePixel");
    io_XWhitePixel();
    break;
```

```
case IO_XDEFAULTGC_CODE:
  io_request_type("XDefaultGC");
  io_XDefaultGC();
  break;

case IO_XDEFAULTROOTWINDOW_CODE:
  io_request_type("XDefaultRootWindow");
  io_XDefaultRootWindow();
  break;

case IO_XDEFAULTSCREEN_CODE:
  io_request_type("XDefaultScreen");
  io_XDefaultScreen();
  break;

case IO_XCLOSEDISPLAY_CODE:
  io_request_type("XCloseDisplay");
  io_XCloseDisplay();
  break;

case IO_XDEFAULTVISUAL_CODE:
  io_request_type("XDefaultVisual");
  io_XDefaultVisual();
  break;

case IO_XFLUSH_CODE:
  io_request_type("XFlush");
  io_XFlush();
  break;

case IO_XCREATEGC_CODE:
  io_request_type("XCreateGC");
  io_XCreateGC();
  break;

case IO_XDEFAULTCOLORMAP_CODE:
  io_request_type("XDefaultColormap");
  io_XDefaultColormap();
  break;

case IO_XDEFAULTDEPTH_CODE:
  io_request_type("XDefaultDepth");
  io_XDefaultDepth();
  break;
```

```
case IO_XFREEGC_CODE:
  io_request_type("XFreeGC");
  io_XFreeGC();
  break;


case IO_XCREATESIMPLEWINDOW_CODE:
  io_request_type("XCreateSimpleWindow");
  io_XCreateSimpleWindow();
  break;

case IO_XDESTROYWINDOW_CODE:
  io_request_type("XDestroyWindow");
  io_XDestroyWindow();
  break;

case IO_XMAPWINDOW_CODE:
  io_request_type("XMapWindow");
  io_XMapWindow();
  break;

case IO_XSETWINDOWBACKGROUND_CODE:
  io_request_type("XSetWindowBackground");
  io_XSetWindowBackground();
  break;

case IO_XUNMAPWINDOW_CODE:
  io_request_type("XUnmapWindow");
  io_XUnmapWindow();
  break;

case IO_XMOVEWINDOW_CODE:
  io_request_type("XMoveWindow");
  io_XMoveWindow();
  break;

case IO_XRESIZEWINDOW_CODE:
  io_request_type("XResizeWindow");
  io_XResizeWindow();
  break;
```

```
case IO_XCLEARAREA_CODE:
  io_request_type("XClearArea");
  io_XClearArea();
  break;

case IO_XCLEARWINDOW_CODE:
  io_request_type("XClearWindow");
  io_XClearWindow();
  break;

case IO_XDRAWPOINT_CODE:
  io_request_type("XDrawPoint");
  io_XDrawPoint();
  break;

case IO_XDRAWPOINTS_CODE:
  io_request_type("XDrawPoints");
  io_XDrawPoints();
  break;

case IO_XDRAWLINE_CODE:
  io_request_type("XDrawLine");
  io_XDrawLine();
  break;

case IO_XDRAWLINES_CODE:
  io_request_type("XDrawLines");
  io_XDrawLines();
  break;

case IO_XDRAWRECTANGLE_CODE:
  io_request_type("XDrawRectangle");
  io_XDrawRectangle();
  break;

case IO_XDRAWARC_CODE:
  io_request_type("XDrawArc");
  io_XDrawArc();
  break;

case IO_XFILLRECTANGLE_CODE:
  io_request_type("XFillRectangle");
  io_XFillRectangle();
  break;
```

```
case IO_XFILLPOLYGON_CODE:
  io_request_type("XFillPolygon");
  io_XFillPolygon();
  break;

case IO_XFILLARC_CODE:
  io_request_type("XFillArc");
  io_XFillArc();
  break;

case IO_XDRAWSTRING_CODE:
  io_request_type("XDrawString");
  io_XDrawString();
  break;

case IO_XDRAWIMAGESTRING_CODE:
  io_request_type("XDrawImageString");
  io_XDrawImageString();
  break;

case IO_XSETBACKGROUND_CODE:
  io_request_type("XSetBackground");
  io_XSetBackground();
  break;

case IO_XSETFOREGROUND_CODE:
  io_request_type("XSetForeground");
  io_XSetForeground();
  break;

case IO_XSETFUNCTION_CODE:
  io_request_type("XSetFunction");
  io_XSetFunction();
  break;

case IO_XSETLINEATTRIBUTES_CODE:
  io_request_type("XSetLineAttributes");
  io_XSetLineAttributes();
  break;



case IO_XSELECTINPUT_CODE:
```

```
  io_request_type("XSelectInput");
  io_XSelectInput();
  break;

case IO_XNEXTEVENT_CODE:
  io_request_type("XNextEvent");
  io_XNextEvent();
  break;

case IO_XCHECKTYPEDEVENT_CODE:
  io_request_type("XCheckTypedEvent");
  io_XCheckTypedEvent();
  break;

case IO_XQUERYPOINTER_CODE:
  io_request_type("XQueryPointer");
  io_XQueryPointer();
  break;

case IO_XSENDEVENT_CODE:
  io_request_type("XSendEvent");
  io_XSendEvent();
  break;


case IO_XALLOCNAMEDCOLOR_CODE:
  io_request_type("XAllocNamedColor");
  io_XAllocNamedColor();
  break;

case IO_XALLOCCOLOR_CODE:
  io_request_type("XAllocColor");
  io_XAllocColor();
  break;

case IO_XCREATECOLORMAP_CODE:
  io_request_type("XCreateColormap");
  io_XCreateColormap();
  break;

case IO_XSETWINDOWCOLORMAP_CODE:
  io_request_type("XSetWindowColormap");
  io_XSetWindowColormap();
  break;
```

```
case IO_XFREECOLORMAP_CODE:
  io_request_type("XFreeColormap");
  io_XFreeColormap();
  break;

case IO_XLOADQUERYFONT_CODE:
  io_request_type("XLoadQueryFont");
  io_XLoadQueryFont();
  break;

case IO_XFREEFONT_CODE:
  io_request_type("XFreeFont");
  io_XFreeFont();
  break;

case IO_XSETFONT_CODE:
  io_request_type("XSetFont");
  io_XSetFont();
  break;


case IO_XCREATEPIXMAP_CODE:
  io_request_type("XCreatePixmap");
  io_XCreatePixmap();
  break;

case IO_XFREEPIXMAP_CODE:
  io_request_type("XFreePixmap");
  io_XFreePixmap();
  break;

case IO_XCOPYAREA_CODE:
  io_request_type("XCopyArea");
  io_XCopyArea();
  break;



case IO_WAITFORMAP_CODE:
  io_request_type("WaitForMap");
  io_WaitForMap();
  break;
```

```
  case IO_DISPLAYARRAY_CODE:
    io_request_type("DisplayArray");
    io_DisplayArray();
    break;


  case IO_XSYNCHRONIZE_CODE:
    io_request_type("XSynchronize");
    io_XSynchronize();
    break;


  default:
    io_request_type("Graphics Invalid");
    io_invalid_graphics_op(graphics_op);
    io_continue_p = FALSE;
    break;
  }
}

/*** io_XOpenDisplay - open a display connection.
 ***
 *** Assumes:
 ***    IO_DATA1   - count  (length of display string)
 ***    IO_DATA2...- display string
 ***
 ***/
VOID io_XOpenDisplay(VOID)
{
  auINT32 count = io_host_ioreq_ptr[IO_DATA1].ls;
  MPI_INT64 *display_buf;
  Display *disp;

  display_buf = io_read_from_monsoon_sbuf(io_host_ioreq_ptr,
    ceiling_uint(count, 8, NULL_uINT32),
    IO_SBUF_IN_OH);

  /* Need to  null-terminate the string */
  ((CHARACTER *) display_buf)[count] = 0;

  disp = XOpenDisplay(((CHARACTER *) display_buf));
  free(display_buf);
  io_write_return_values((auINT32) disp, 0);
```

```
  /* Add display to list that keeps track of these.
   */
  IO_Displays = io_cons (disp, IO_Displays);

}

/*** io_XBlackPixel - get the pixel value of a black pixel.
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - screen_number
 ***/
VOID io_XBlackPixel(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  auINT32 screen_number = io_host_ioreq_ptr[IO_DATA2].ls;

  io_write_return_values(XBlackPixel(disp, screen_number), 0);
}


/*** io_XWhitePixel - get the pixel value of a white pixel.
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - screen_number
 ***/
VOID io_XWhitePixel(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  auINT32 screen_number = io_host_ioreq_ptr[IO_DATA2].ls;

  io_write_return_values(XWhitePixel(disp, screen_number),0);
}

/*** io_XDefaultGC - get the default graphics context.
 ***
 *** Assumes:
 ***    IO_DATA1 - disp
 ***    IO_DATA2 - screen_number
 ***/
VOID io_XDefaultGC(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
```

```
  auINT32 screen_number = io_host_ioreq_ptr[IO_DATA2].ls;

  io_write_return_values((auINT32) XDefaultGC(disp, screen_number),0);
}

/*** io_XDefaultRootWindow - get the default graphics context.
 ***
 *** Assumes:
 ***    IO_DATA1 - disp
 ***/
VOID io_XDefaultRootWindow(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;

  io_write_return_values(XDefaultRootWindow(disp),0);
}

/*** io_XDefaultScreen - get the default screen number.
 ***
 *** Assumes:
 ***    IO_DATA1 - disp
 ***/
VOID io_XDefaultScreen(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;

  io_write_return_values(XDefaultScreen(disp),0);
}

/*** io_XCloseDisplay - get the default screen number.
 ***
 *** Assumes:
 ***    IO_DATA1 - disp
 ***/
VOID io_XCloseDisplay(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;

  XCloseDisplay(disp);
}

/*** io_XDefaultVisual - get the default visual.
 ***
 *** Assumes:
```

```
 ***    IO_DATA1 - disp
 ***    IO_DATA2 - screen_number
 ***/
VOID io_XDefaultVisual(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  auINT32 screen_number = io_host_ioreq_ptr[IO_DATA2].ls;

  io_write_return_values((auINT32) XDefaultVisual(disp, screen_number),0);
}

/*** io_XFlush - flush the request buffer.
 ***
 *** Assumes:
 ***    IO_DATA1 - disp
 ***/
VOID io_XFlush(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;

  XFlush(disp);
}

/*** io_XCreateGC - create a new graphics context (with default values).
 ***
 *** Assumes:
 ***    IO_DATA1 - disp
 ***    IO_DATA2 - drawable
 ***/
VOID io_XCreateGC(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  auINT32 drawable = io_host_ioreq_ptr[IO_DATA2].ls;
  XGCValues xgc;

  io_write_return_values((auINT32) XCreateGC(disp, drawable, 0, &xgc),0);
}

/*** io_XDefaultColormap - get the default colormap.
 ***
 *** Assumes:
 ***    IO_DATA1 - disp
 ***    IO_DATA2 - screen_number
 ***/
```

```
VOID io_XDefaultColormap(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  auINT32 screen_number = io_host_ioreq_ptr[IO_DATA2].ls;

  io_write_return_values((auINT32) XDefaultColormap(disp, screen_number),0);
}

/*** io_XDefaultDepth - get the default depth.
 ***
 *** Assumes:
 ***    IO_DATA1 - disp
 ***    IO_DATA2 - screen_number
 ***/
VOID io_XDefaultDepth(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  auINT32 screen_number = io_host_ioreq_ptr[IO_DATA2].ls;

  io_write_return_values((auINT32) XDefaultDepth(disp, screen_number),0);
}

/*** io_XFreeGC - free a graphics context.
 ***
 *** Assumes:
 ***    IO_DATA1 - disp
 ***    IO_DATA2 - gc
 ***/
VOID io_XFreeGC(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA2].ls;

  XFreeGC(disp, gc);
}



/*** io_XCreateSimpleWindow - create a simple window.
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - parent
 ***    IO_DATA3 - x
```

```
 ***    IO_DATA4 - y
 ***    IO_DATA5 - width
 ***    IO_DATA6 - height
 ***    IO_DATA7 - border_width
 ***    IO_DATA8 - border
 ***    IO_DATA9 - background
 ***/
VOID io_XCreateSimpleWindow(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Window parent = (Window) io_host_ioreq_ptr[IO_DATA2].ls;
  auINT32 x = io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 y = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 width = io_host_ioreq_ptr[IO_DATA5].ls;
  auINT32 height = io_host_ioreq_ptr[IO_DATA6].ls;
  auINT32 border_width = io_host_ioreq_ptr[IO_DATA7].ls;
  auINT32 border = io_host_ioreq_ptr[IO_DATA8].ls;
  auINT32 background = io_host_ioreq_ptr[IO_DATA9].ls;
  Window retval;

  retval = XCreateSimpleWindow(disp, parent, x, y, width, height,
       border_width, border, background);

  io_write_return_values(retval, 0);
}

/*** io_XDestroyWindow - destroy a window
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - window
 ***/
VOID io_XDestroyWindow(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Window win = (Window) io_host_ioreq_ptr[IO_DATA2].ls;

  XDestroyWindow(disp, win);
}


/*** io_XMapWindow - map a window on a given display
 ***
 *** Assumes:
```

```
 ***    IO_DATA1 - display
 ***    IO_DATA2 - window
 ***/
VOID io_XMapWindow(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Window win = (Window) io_host_ioreq_ptr[IO_DATA2].ls;

  XMapWindow(disp, win);
}


/*** io_XSetWindowBackground - set the background pixel of a window
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - window
 ***    IO_DATA3 - pixel
 ***/
VOID io_XSetWindowBackground(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Window win = (Window) io_host_ioreq_ptr[IO_DATA2].ls;
  auINT32 pixel = io_host_ioreq_ptr[IO_DATA3].ls;

  XSetWindowBackground(disp, win, pixel);
}

/*** io_XUnmapWindow - unmap a window on a given display
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - window
 ***/
VOID io_XUnmapWindow(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Window win = (Window) io_host_ioreq_ptr[IO_DATA2].ls;

  XUnmapWindow(disp, win);
}

/*** io_XMoveWindow - move a window.
 ***
```

```
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - drawable
 ***    IO_DATA3 - x
 ***    IO_DATA4 - y
 ***/
VOID io_XMoveWindow(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Drawable drawable = (Drawable) io_host_ioreq_ptr[IO_DATA2].ls;
  auINT32 x = io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 y = io_host_ioreq_ptr[IO_DATA4].ls;

  XMoveWindow(disp, drawable, x, y);
}

/*** io_XResizeWindow - resize a window.
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - drawable
 ***    IO_DATA3 - x
 ***    IO_DATA4 - y
 ***/
VOID io_XResizeWindow(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Drawable drawable = (Drawable) io_host_ioreq_ptr[IO_DATA2].ls;
  auINT32 x = io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 y = io_host_ioreq_ptr[IO_DATA4].ls;

  XResizeWindow(disp, drawable, x, y);
}



/*** io_XClearArea - clear an area of a window to the background pixel
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - window
 ***    IO_DATA3 - x
 ***    IO_DATA4 - y
 ***    IO_DATA5 - width
```

```
 ***    IO_DATA6 - height
 ***    IO_DATA7 - exposures_c_bool
 ***/
VOID io_XClearArea(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Window win = (Window) io_host_ioreq_ptr[IO_DATA2].ls;
  auINT32 x = io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 y = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 width = io_host_ioreq_ptr[IO_DATA5].ls;
  auINT32 height = io_host_ioreq_ptr[IO_DATA6].ls;
  auINT32 exposures_c_bool = io_host_ioreq_ptr[IO_DATA7].ls;

  XClearArea(disp, win, x, y, width, height, exposures_c_bool);
}

/*** io_XClearWindow - clear the contents of a window.
 ***
 *** Assumes:
 ***   IO_DATA1 - display
 ***   IO_DATA2 - window
 ***/
VOID io_XClearWindow(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Window win = (Window) io_host_ioreq_ptr[IO_DATA2].ls;

  XClearWindow(disp, win);
}

/*** io_XDrawPoint - draw a single point.
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - drawable
 ***    IO_DATA3 - gc
 ***    IO_DATA4 - x
 ***    IO_DATA5 - y
 ***/
VOID io_XDrawPoint(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Drawable drawable = (Drawable) io_host_ioreq_ptr[IO_DATA2].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA3].ls;
```

```
  auINT32 x = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 y = io_host_ioreq_ptr[IO_DATA5].ls;

  XDrawPoint(disp, drawable, gc, x, y);
}

/*** io_XDrawPoints - draw multiple points
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - drawable
 ***    IO_DATA3 - gc
 ***    IO_DATA4 - number of points
 ***    IO_DATA5 - mode : 0 = CoordModeOrigin, !0 = CoordModePrevious
 ***    IO_BUF   - array of points (two 64-bits words per point)
 ***/
VOID io_XDrawPoints(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Drawable drawable = (Drawable) io_host_ioreq_ptr[IO_DATA2].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 npoints = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 mode = io_host_ioreq_ptr[IO_DATA5].ls;
  XPoint *points = (XPoint*)malloc(npoints*sizeof(XPoint));
  MPI_INT64 *string_buf;
  auINT32 i;

  string_buf = io_read_from_monsoon_sbuf(
                   io_host_ioreq_ptr, npoints*2, IO_SBUF_IN_OH);
  for( i=0 ; i<npoints ; i++ )
    {
     points[i].x = string_buf[2*i].ls;
     points[i].y = string_buf[2*i+1].ls;
    }
  if (mode)
       XDrawPoints(disp, drawable, gc, points, npoints, CoordModePrevious);
  else XDrawPoints(disp, drawable, gc, points, npoints, CoordModeOrigin);
  free(string_buf);
  free(points);
}

/*** io_XDrawLine - draw a line
 ***
 *** Assumes:
```

```
 ***    IO_DATA1 - display
 ***    IO_DATA2 - drawable
 ***    IO_DATA3 - gc
 ***    IO_DATA4 - x1
 ***    IO_DATA5 - y1
 ***    IO_DATA6 - x2
 ***    IO_DATA7 - y2
 ***/
VOID io_XDrawLine(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Drawable drawable = (Drawable) io_host_ioreq_ptr[IO_DATA2].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 x1 = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 y1 = io_host_ioreq_ptr[IO_DATA5].ls;
  auINT32 x2 = io_host_ioreq_ptr[IO_DATA6].ls;
  auINT32 y2 = io_host_ioreq_ptr[IO_DATA7].ls;

  XDrawLine(disp, drawable, gc, x1, y1, x2, y2);
}

/*** io_XDrawLines - draw multiple connected lines
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - drawable
 ***    IO_DATA3 - gc
 ***    IO_DATA4 - number of points
 ***    IO_DATA5 - mode : 0 = CoordModeOrigin, !0 = CoordModePrevious
 ***    IO_BUF   - array of points (two 64-bits words per point)
 ***/
VOID io_XDrawLines(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Drawable drawable = (Drawable) io_host_ioreq_ptr[IO_DATA2].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 npoints = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 mode = io_host_ioreq_ptr[IO_DATA5].ls;
  XPoint *points = (XPoint*)malloc(npoints*sizeof(XPoint));
  MPI_INT64 *string_buf;
  auINT32 i;

  string_buf = io_read_from_monsoon_sbuf(
                    io_host_ioreq_ptr, npoints*2, IO_SBUF_IN_OH);
```

```
  for( i=0 ; i<npoints ; i++ )
    {
     points[i].x = string_buf[2*i].ls;
     points[i].y = string_buf[2*i+1].ls;
    }
  if (mode)
       XDrawLines(disp, drawable, gc, points, npoints,  CoordModePrevious);
  else XDrawLines(disp, drawable, gc, points, npoints, CoordModeOrigin);
  free(string_buf);
  free(points);
}

/*** io_XDrawRectangle - draw a rectangle
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - drawable
 ***    IO_DATA3 - gc
 ***    IO_DATA4 - x
 ***    IO_DATA5 - y
 ***    IO_DATA6 - width
 ***    IO_DATA7 - height
 ***/
VOID io_XDrawRectangle(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Drawable drawable = (Drawable) io_host_ioreq_ptr[IO_DATA2].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 x = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 y = io_host_ioreq_ptr[IO_DATA5].ls;
  auINT32 width = io_host_ioreq_ptr[IO_DATA6].ls;
  auINT32 height = io_host_ioreq_ptr[IO_DATA7].ls;

  XDrawRectangle(disp, drawable, gc, x, y, width, height);
}

/*** io_XDrawArc - draw an arc
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - drawable
 ***    IO_DATA3 - gc
 ***    IO_DATA4 - x
 ***    IO_DATA5 - y
```

```
 ***    IO_DATA6 - width
 ***    IO_DATA7 - height
 ***    IO_DATA8 - angle1
 ***    IO_DATA9 - angle2
 ***/
VOID io_XDrawArc(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Drawable drawable = (Drawable) io_host_ioreq_ptr[IO_DATA2].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 x = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 y = io_host_ioreq_ptr[IO_DATA5].ls;
  auINT32 width = io_host_ioreq_ptr[IO_DATA6].ls;
  auINT32 height = io_host_ioreq_ptr[IO_DATA7].ls;
  auINT32 angle1 = io_host_ioreq_ptr[IO_DATA8].ls;
  auINT32 angle2 = io_host_ioreq_ptr[IO_DATA9].ls;

  XDrawArc(disp, drawable, gc, x, y, width, height, angle1, angle2);
}

/*** io_XFillRectangle - fill a rectangle
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - drawable
 ***    IO_DATA3 - gc
 ***    IO_DATA4 - x
 ***    IO_DATA5 - y
 ***    IO_DATA6 - width
 ***    IO_DATA7 - height
 ***/
VOID io_XFillRectangle(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Drawable drawable = (Drawable) io_host_ioreq_ptr[IO_DATA2].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 x = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 y = io_host_ioreq_ptr[IO_DATA5].ls;
  auINT32 width = io_host_ioreq_ptr[IO_DATA6].ls;
  auINT32 height = io_host_ioreq_ptr[IO_DATA7].ls;

  XFillRectangle(disp, drawable, gc, x, y, width, height);
}
```

```
/*** io_XFillPolygon - fill a polygon
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - drawable
 ***    IO_DATA3 - gc
 ***    IO_DATA4 - number of points
 ***    IO_DATA5 - mode : 0 = CoordModeOrigin, !0 = CoordModePrevious
 ***    IO_BUF   - array of points (two 64-bits words per point)
 ***/
VOID io_XFillPolygon(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Drawable drawable = (Drawable) io_host_ioreq_ptr[IO_DATA2].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 npoints = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 mode = io_host_ioreq_ptr[IO_DATA5].ls;
  XPoint *points = (XPoint*)malloc(npoints*sizeof(XPoint));
  MPI_INT64 *string_buf;
  auINT32 i;

  string_buf = io_read_from_monsoon_sbuf(
                   io_host_ioreq_ptr, npoints*2, IO_SBUF_IN_OH);
  for( i=0 ; i<npoints ; i++ )
    {
     points[i].x = string_buf[2*i].ls;
     points[i].y = string_buf[2*i+1].ls;
    }
  if (mode)
       XFillPolygon(disp, drawable, gc,
           points, npoints, Complex, CoordModePrevious);
  else XFillPolygon(disp, drawable, gc,
           points, npoints, Complex, CoordModeOrigin);
  free(string_buf);
  free(points);
}

/*** io_XFillArc - fill an arc
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - drawable
 ***    IO_DATA3 - gc
 ***    IO_DATA4 - x
```

```
 ***    IO_DATA5 - y
 ***    IO_DATA6 - width
 ***    IO_DATA7 - height
 ***    IO_DATA8 - angle1
 ***    IO_DATA9 - angle2
 ***/
VOID io_XFillArc(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Drawable drawable = (Drawable) io_host_ioreq_ptr[IO_DATA2].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 x = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 y = io_host_ioreq_ptr[IO_DATA5].ls;
  auINT32 width = io_host_ioreq_ptr[IO_DATA6].ls;
  auINT32 height = io_host_ioreq_ptr[IO_DATA7].ls;
  auINT32 angle1 = io_host_ioreq_ptr[IO_DATA8].ls;
  auINT32 angle2 = io_host_ioreq_ptr[IO_DATA9].ls;

  XFillArc(disp, drawable, gc, x, y, width, height, angle1, angle2);
}

/*** io_XDrawString - draw a string of characters.
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - drawable
 ***    IO_DATA3 - gc
 ***    IO_DATA4 - x
 ***    IO_DATA5 - y
 ***    IO_DATA6 - length of string
 ***    IO_BUFFER - string
 ***/
VOID io_XDrawString(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Drawable drawable = (Drawable) io_host_ioreq_ptr[IO_DATA2].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 x = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 y = io_host_ioreq_ptr[IO_DATA5].ls;
  auINT32 length = io_host_ioreq_ptr[IO_DATA6].ls;
  auINT32 nwords = ceiling_uint(length, 8, NULL_uINT32);
  MPI_INT64 *string_buf;

  string_buf = io_read_from_monsoon_sbuf(io_host_ioreq_ptr,
```

```
    nwords,
    IO_SBUF_IN_OH);

  ((CHARACTER *) string_buf)[length] = 0;

  XDrawString(disp, drawable, gc, x, y, ((CHARACTER *) string_buf), length);
  free(string_buf);
}

/*** io_XDrawImageString - draw a string of characters.
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - drawable
 ***    IO_DATA3 - gc
 ***    IO_DATA4 - x
 ***    IO_DATA5 - y
 ***    IO_DATA6 - length of string
 ***    IO_BUFFER - string
 ***/
VOID io_XDrawImageString(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;

  Drawable drawable = (Drawable) io_host_ioreq_ptr[IO_DATA2].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 x = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 y = io_host_ioreq_ptr[IO_DATA5].ls;
  auINT32 length = io_host_ioreq_ptr[IO_DATA6].ls;
  auINT32 nwords = ceiling_uint(length, 8, NULL_uINT32);
  MPI_INT64 *string_buf;

  string_buf = io_read_from_monsoon_sbuf(io_host_ioreq_ptr,
    nwords,
    IO_SBUF_IN_OH);

  ((CHARACTER *) string_buf)[length] = 0;

  XDrawImageString(disp, drawable, gc, x, y,
                       ((CHARACTER *) string_buf), length);
  free(string_buf);
}

/*** io_XSetBackground - set the background pixel value in a graphics context
```

```
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - gc
 ***    IO_DATA3 - pixel
 ***/
VOID io_XSetBackground(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA2].ls;
  auINT32 pixel = io_host_ioreq_ptr[IO_DATA3].ls;

  XSetBackground(disp, gc, pixel);
}

/*** io_XSetForeground - set the foreground pixel value in a graphics context
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - gc
 ***    IO_DATA3 - pixel
 ***/
VOID io_XSetForeground(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA2].ls;
  auINT32 pixel = io_host_ioreq_ptr[IO_DATA3].ls;

  XSetForeground(disp, gc, pixel);
}

/*** io_XSetFunction - set the bitwise logical operation in a graphics context
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - gc
 ***    IO_DATA3 - function
 ***/
VOID io_XSetFunction(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA2].ls;
  auINT32 function = io_host_ioreq_ptr[IO_DATA3].ls;
```

```
  XSetFunction(disp, gc, function);
}

/*** io_XSetLineAttributes - set the line drawing
                             components in a graphics context.
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - gc
 ***    IO_DATA3 - line_width
 ***    IO_DATA4 - line_style
 ***    IO_DATA5 - cap_style
 ***    IO_DATA6 - join_style
 ***/
VOID io_XSetLineAttributes(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA2].ls;
  auINT32 line_width = io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 line_style = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 cap_style = io_host_ioreq_ptr[IO_DATA5].ls;
  auINT32 join_style = io_host_ioreq_ptr[IO_DATA6].ls;

  XSetLineAttributes(disp, gc, line_width, line_style, cap_style, join_style);
}




/*** io_XSelectInput - select the event types to be sent to a window
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - window
 ***    IO_DATA3 - mask_event
 ***/
VOID io_XSelectInput(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Window win = (Window) io_host_ioreq_ptr[IO_DATA2].ls;
  auINT32 mask_event = io_host_ioreq_ptr[IO_DATA3].ls;

  XSelectInput(disp, win, mask_event);
```

```
}


VOID copy_event(XEvent *ev)
{
  MPI_INT64 param_buf[10];
  auINT32 count,asckey;
  char    c;
  KeySym  ks;

  switch(ev->type) {
  case KeyPress :
    if(XLookupString((XKeyEvent*)ev,&c,1,&ks,NULL)!=0) asckey=c;
    else asckey=0;
    asckey&=255;
    param_buf[0]=(MPI_INT64){0,ev->xkey.window};
    param_buf[1]=(MPI_INT64){0,ev->xkey.x};
    param_buf[2]=(MPI_INT64){0,ev->xkey.y};
    param_buf[3]=(MPI_INT64){0,ev->xkey.x_root};
    param_buf[4]=(MPI_INT64){0,ev->xkey.y_root};
    param_buf[5]=(MPI_INT64){0,ev->xkey.state};
    param_buf[6]=(MPI_INT64){0,ev->xkey.keycode};
    param_buf[7]=(MPI_INT64){0,asckey};
    count=8;
    break;
  case KeyRelease :
    if(XLookupString((XKeyEvent*)ev,&c,1,&ks,NULL)!=0) asckey=c;
    else asckey=0;
    asckey&=255;
    param_buf[0]=(MPI_INT64){0,ev->xkey.window};
    param_buf[1]=(MPI_INT64){0,ev->xkey.x};
    param_buf[2]=(MPI_INT64){0,ev->xkey.y};
    param_buf[3]=(MPI_INT64){0,ev->xkey.x_root};
    param_buf[4]=(MPI_INT64){0,ev->xkey.y_root};
    param_buf[5]=(MPI_INT64){0,ev->xkey.state};
    param_buf[6]=(MPI_INT64){0,ev->xkey.keycode};
    param_buf[7]=(MPI_INT64){0,asckey};
    count=8;
    break;
  case ButtonPress :
    param_buf[0]=(MPI_INT64){0,ev->xbutton.window};
    param_buf[1]=(MPI_INT64){0,ev->xbutton.x};
    param_buf[2]=(MPI_INT64){0,ev->xbutton.y};
    param_buf[3]=(MPI_INT64){0,ev->xbutton.x_root};
```

```
  param_buf[4]=(MPI_INT64){0,ev->xbutton.y_root};
  param_buf[5]=(MPI_INT64){0,ev->xbutton.state};
  param_buf[6]=(MPI_INT64){0,ev->xbutton.button};
  count=7;
  break;
case ButtonRelease :
  param_buf[0]=(MPI_INT64){0,ev->xbutton.window};
  param_buf[1]=(MPI_INT64){0,ev->xbutton.x};
  param_buf[2]=(MPI_INT64){0,ev->xbutton.y};
  param_buf[3]=(MPI_INT64){0,ev->xbutton.x_root};
  param_buf[4]=(MPI_INT64){0,ev->xbutton.y_root};
  param_buf[5]=(MPI_INT64){0,ev->xbutton.state};
  param_buf[6]=(MPI_INT64){0,ev->xbutton.button};
  count=7;
  break;
case MotionNotify :
  param_buf[0]=(MPI_INT64){0,ev->xmotion.window};
  param_buf[1]=(MPI_INT64){0,ev->xmotion.x};
  param_buf[2]=(MPI_INT64){0,ev->xmotion.y};
  param_buf[3]=(MPI_INT64){0,ev->xmotion.x_root};
  param_buf[4]=(MPI_INT64){0,ev->xmotion.y_root};
  param_buf[5]=(MPI_INT64){0,ev->xmotion.state};
  count=6;
  break;
case EnterNotify :
  param_buf[0]=(MPI_INT64){0,ev->xcrossing.window};
  param_buf[1]=(MPI_INT64){0,ev->xcrossing.x};
  param_buf[2]=(MPI_INT64){0,ev->xcrossing.y};
  param_buf[3]=(MPI_INT64){0,ev->xcrossing.x_root};
  param_buf[4]=(MPI_INT64){0,ev->xcrossing.y_root};
  param_buf[5]=(MPI_INT64){0,ev->xcrossing.state};
  count=6;
  break;
case LeaveNotify :
  param_buf[0]=(MPI_INT64){0,ev->xcrossing.window};
  param_buf[1]=(MPI_INT64){0,ev->xcrossing.x};
  param_buf[2]=(MPI_INT64){0,ev->xcrossing.y};
  param_buf[3]=(MPI_INT64){0,ev->xcrossing.x_root};
  param_buf[4]=(MPI_INT64){0,ev->xcrossing.y_root};
  param_buf[5]=(MPI_INT64){0,ev->xcrossing.state};
  count=6;
  break;
case Expose :
  param_buf[0]=(MPI_INT64){0,ev->xexpose.window};
```

```
    param_buf[1]=(MPI_INT64){0,ev->xexpose.x};
    param_buf[2]=(MPI_INT64){0,ev->xexpose.y};
    param_buf[3]=(MPI_INT64){0,ev->xexpose.width};
    param_buf[4]=(MPI_INT64){0,ev->xexpose.height};
    count=5;
    break;
  case MapNotify :
    param_buf[0]=(MPI_INT64){0,ev->xmap.window};
    count=1;
    break;
  case ConfigureNotify :
    param_buf[0]=(MPI_INT64){0,ev->xconfigure.window};
    param_buf[1]=(MPI_INT64){0,ev->xconfigure.x};
    param_buf[2]=(MPI_INT64){0,ev->xconfigure.y};
    param_buf[3]=(MPI_INT64){0,ev->xconfigure.width};
    param_buf[4]=(MPI_INT64){0,ev->xconfigure.height};
    count=5;
    break;
  default :
    param_buf[0]=(MPI_INT64){0,ev->type};
    count=1;
    break;}
  io_write_to_monsoon_sbuf(
        io_host_ioreq_ptr, param_buf, count , IO_SBUF_IN_OH);
  io_write_return_values((auINT32) ev->type, 0);
}

/*** io_XNextEvent - get the next event of any type or window
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 *** Return the type, and fill the buffer with the useful parameters
 ***/
VOID io_XNextEvent(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  XEvent ev;

  XNextEvent(disp , &ev);
  copy_event(&ev);
}

/*** io_XCheckTypedEvent - return the next event in the queue that
                           matches event type ; don't wait
```

```
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - event_type
 *** Return the type, and fill the buffer with the useful parameters
 ***/
VOID io_XCheckTypedEvent(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  auINT32 event_type = io_host_ioreq_ptr[IO_DATA2].ls;
  XEvent ev;

  if (!XCheckTypedEvent(disp, event_type, &ev)) ev.type = 0;
  copy_event(&ev);
}

/*** io_XQueryPointer - get the current pointer location
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - window
 *** Return keys_buttons, and fill the buffer with the useful parameters
 ***/
VOID io_XQueryPointer(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Window win = (Window) io_host_ioreq_ptr[IO_DATA2].ls;

  MPI_INT64 param_buf[6];
  Window root, child;
  int root_x, root_y, win_x, win_y;
  unsigned int keys_buttons;

  XQueryPointer(disp, win, &root, &child,
                &root_x, &root_y, &win_x, &win_y, &keys_buttons);

  param_buf[0]=(MPI_INT64){0,root};
  param_buf[1]=(MPI_INT64){0,child};
  param_buf[2]=(MPI_INT64){0,root_x};
  param_buf[3]=(MPI_INT64){0,root_y};
  param_buf[4]=(MPI_INT64){ ((win_x<0)?-1:0) ,win_x};
  param_buf[5]=(MPI_INT64){ ((win_y<0)?-1:0) ,win_y};
  io_write_to_monsoon_sbuf(io_host_ioreq_ptr, param_buf, 6, IO_SBUF_IN_OH);
  io_write_return_values((auINT32) keys_buttons, 0);
```

```
}

/*** io_XSendEvent - send a event.
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - window
 ***    IO_DATA3 - event_mask
 ***    IO_DATA4 - type
 ***    IO_DATA. - others arguments (different for each type of event)
 ***/
VOID io_XSendEvent(VOID)
{
  XEvent ev;
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Window win = (Window) io_host_ioreq_ptr[IO_DATA2].ls;
  auINT32 event_mask = io_host_ioreq_ptr[IO_DATA3].ls;

  ev.type = (int) io_host_ioreq_ptr[IO_DATA4].ls;
  ev.xany.display = disp;
  ev.xany.window = win;

  switch(ev.type) {
  case KeyPress :
    ev.xkey.x = (int) io_host_ioreq_ptr[IO_DATA5].ls;
    ev.xkey.y = (int) io_host_ioreq_ptr[IO_DATA6].ls;
    ev.xkey.x_root = (int) io_host_ioreq_ptr[IO_DATA7].ls;
    ev.xkey.y_root = (int) io_host_ioreq_ptr[IO_DATA8].ls;
    ev.xkey.state = (unsigned int) io_host_ioreq_ptr[IO_DATA9].ls;
    ev.xkey.keycode = (unsigned int) io_host_ioreq_ptr[IO_DATA10].ls;
    break;
  case KeyRelease :
    ev.xkey.x = (int) io_host_ioreq_ptr[IO_DATA5].ls;
    ev.xkey.y = (int) io_host_ioreq_ptr[IO_DATA6].ls;
    ev.xkey.x_root = (int) io_host_ioreq_ptr[IO_DATA7].ls;
    ev.xkey.y_root = (int) io_host_ioreq_ptr[IO_DATA8].ls;
    ev.xkey.state = (unsigned int) io_host_ioreq_ptr[IO_DATA9].ls;
    ev.xkey.keycode = (unsigned int) io_host_ioreq_ptr[IO_DATA10].ls;
    break;
  case ButtonPress :
    ev.xbutton.x = (int) io_host_ioreq_ptr[IO_DATA5].ls;
    ev.xbutton.y = (int) io_host_ioreq_ptr[IO_DATA6].ls;
    ev.xbutton.x_root = (int) io_host_ioreq_ptr[IO_DATA7].ls;
    ev.xbutton.y_root = (int) io_host_ioreq_ptr[IO_DATA8].ls;
```

```
  ev.xbutton.state = (unsigned int) io_host_ioreq_ptr[IO_DATA9].ls;
  ev.xbutton.button = (unsigned int) io_host_ioreq_ptr[IO_DATA10].ls;
  break;
case ButtonRelease :
  ev.xbutton.x = (int) io_host_ioreq_ptr[IO_DATA5].ls;
  ev.xbutton.y = (int) io_host_ioreq_ptr[IO_DATA6].ls;
  ev.xbutton.x_root = (int) io_host_ioreq_ptr[IO_DATA7].ls;
  ev.xbutton.y_root = (int) io_host_ioreq_ptr[IO_DATA8].ls;
  ev.xbutton.state = (unsigned int) io_host_ioreq_ptr[IO_DATA9].ls;
  ev.xbutton.button = (unsigned int) io_host_ioreq_ptr[IO_DATA10].ls;
  break;
case MotionNotify :
  ev.xmotion.x = (int) io_host_ioreq_ptr[IO_DATA5].ls;
  ev.xmotion.y = (int) io_host_ioreq_ptr[IO_DATA6].ls;
  ev.xmotion.x_root = (int) io_host_ioreq_ptr[IO_DATA7].ls;
  ev.xmotion.y_root = (int) io_host_ioreq_ptr[IO_DATA8].ls;
  ev.xmotion.state = (unsigned int) io_host_ioreq_ptr[IO_DATA9].ls;
  break;
case EnterNotify :
  ev.xcrossing.x = (int) io_host_ioreq_ptr[IO_DATA5].ls;
  ev.xcrossing.y = (int) io_host_ioreq_ptr[IO_DATA6].ls;
  ev.xcrossing.x_root = (int) io_host_ioreq_ptr[IO_DATA7].ls;
  ev.xcrossing.y_root = (int) io_host_ioreq_ptr[IO_DATA8].ls;
  ev.xcrossing.state = (unsigned int) io_host_ioreq_ptr[IO_DATA9].ls;
  break;
case LeaveNotify :
  ev.xcrossing.x = (int) io_host_ioreq_ptr[IO_DATA5].ls;
  ev.xcrossing.y = (int) io_host_ioreq_ptr[IO_DATA6].ls;
  ev.xcrossing.x_root = (int) io_host_ioreq_ptr[IO_DATA7].ls;
  ev.xcrossing.y_root = (int) io_host_ioreq_ptr[IO_DATA8].ls;
  ev.xcrossing.state = (unsigned int) io_host_ioreq_ptr[IO_DATA9].ls;
  break;
case Expose :
  ev.xexpose.x = (int) io_host_ioreq_ptr[IO_DATA5].ls;
  ev.xexpose.y = (int) io_host_ioreq_ptr[IO_DATA6].ls;
  ev.xexpose.width = (int) io_host_ioreq_ptr[IO_DATA7].ls;
  ev.xexpose.height = (int) io_host_ioreq_ptr[IO_DATA8].ls;
  break;
case MapNotify :
  break;
case ConfigureNotify :
  ev.xconfigure.x = (int) io_host_ioreq_ptr[IO_DATA5].ls;
  ev.xconfigure.y = (int) io_host_ioreq_ptr[IO_DATA6].ls;
  ev.xconfigure.width = (int) io_host_ioreq_ptr[IO_DATA7].ls;
```

```
    ev.xconfigure.height = (int) io_host_ioreq_ptr[IO_DATA8].ls;
    break;
  default :
    break;}
  XSendEvent(disp, win, False, event_mask, &ev);
}


/*** io_XAllocNamedColor - allocate a read-only colorcell from color name.
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - cmap
 ***    IO_DATA3 - length of string
 ***    IO_BUFFER - string
 ***/
VOID io_XAllocNamedColor(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Colormap cmap= (Colormap) io_host_ioreq_ptr[IO_DATA2].ls;
  auINT32 length = io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 nwords = ceiling_uint(length, 8, NULL_uINT32);
  MPI_INT64 *string_buf;
  XColor tCol,tCol_exact;
  auINT32 color;

  string_buf = io_read_from_monsoon_sbuf(io_host_ioreq_ptr,
    nwords,
    IO_SBUF_IN_OH);

  ((CHARACTER *) string_buf)[length] = 0;
  XAllocNamedColor(disp, cmap, ((CHARACTER *) string_buf), &tCol, &tCol_exact);
  color=tCol.pixel;

  free(string_buf);
  io_write_return_values((auINT32) color, 0);
}

/*** io_XAllocColor - allocate a read-only colormap cell with closest
                      hardware-supported color.
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - cmap
```

```
 ***    IO_DATA3 - red
 ***    IO_DATA4 - green
 ***    IO_DATA5 - blue
 ***/
VOID io_XAllocColor(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Colormap cmap= (Colormap) io_host_ioreq_ptr[IO_DATA2].ls;
  auINT32 red = io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 green = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 blue = io_host_ioreq_ptr[IO_DATA5].ls;
  XColor col;
  auINT32 code_color;

  col.red = red;
  col.green = green;
  col.blue = blue;
  col.pixel=0;
  XAllocColor(disp, cmap, &col);
  code_color=col.pixel;

  io_write_return_values((auINT32) code_color, 0);
}

/*** io_XCreateColormap - create a colormap.
 ***
 *** Assumes:
 ***    IO_DATA1 - disp
 ***    IO_DATA2 - window
 ***    IO_DATA3 - visual
 ***/
VOID io_XCreateColormap(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Window win = (Window) io_host_ioreq_ptr[IO_DATA2].ls;
  Visual *vis = (Visual *) io_host_ioreq_ptr[IO_DATA3].ls;

  io_write_return_values(
        (auINT32) XCreateColormap(disp, win, vis, AllocNone),0);
}

/*** io_XSetWindowColormap - create the colormap attribute for a window.
 ***
 *** Assumes:
```

```
 ***    IO_DATA1 - disp
 ***    IO_DATA2 - window
 ***    IO_DATA3 - cmap
 ***/
VOID io_XSetWindowColormap(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Window win = (Window) io_host_ioreq_ptr[IO_DATA2].ls;
  Colormap cmap = (Colormap) io_host_ioreq_ptr[IO_DATA3].ls;

  XSetWindowColormap(disp, win, cmap);
}

/*** io_XFreeColormap - delete a colormap and install the default colormap.
 ***
 *** Assumes:
 ***    IO_DATA1 - disp
 ***    IO_DATA2 - cmap
 ***/
VOID io_XFreeColormap(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Colormap cmap = (Colormap) io_host_ioreq_ptr[IO_DATA2].ls;

  XFreeColormap(disp, cmap);
}

/*** io_XLoadQueryFont - load a font and fill information structure.
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - length of string
 ***    IO_BUFFER - string
 ***/
VOID io_XLoadQueryFont(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  auINT32 length = io_host_ioreq_ptr[IO_DATA2].ls;
  auINT32 nwords = ceiling_uint(length, 8, NULL_uINT32);
  MPI_INT64 *string_buf;

  XFontStruct *font;

  string_buf = io_read_from_monsoon_sbuf(io_host_ioreq_ptr,
```

```
    nwords,
    IO_SBUF_IN_OH);

  ((CHARACTER *) string_buf)[length] = 0;
  font = XLoadQueryFont(disp, ((CHARACTER *) string_buf));

  free(string_buf);
  io_write_return_values((auINT32) font, 0);
}

/*** io_XFreeFont - unload a font and free storage for the font structure.
 ***
 *** Assumes:
 ***    IO_DATA1 - disp
 ***    IO_DATA2 - font
 ***/
VOID io_XFreeFont(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  XFontStruct *font = (XFontStruct *) io_host_ioreq_ptr[IO_DATA2].ls;

  XFreeFont(disp, font);
}

/*** io_XSetFont - set the current font in a graphics context.
 ***
 *** Assumes:
 ***    IO_DATA1 - disp
 ***    IO_DATA2 - gc
 ***    IO_DATA3 - font
 ***/
VOID io_XSetFont(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA2].ls;
  XFontStruct *font = (XFontStruct *) io_host_ioreq_ptr[IO_DATA3].ls;

  XSetFont(disp, gc, font->fid);
}



/*** io_XCreatePixmap - create a pixmap.
 ***
```

```
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - drawable
 ***    IO_DATA3 - width
 ***    IO_DATA4 - height
 ***    IO_DATA5 - depth
 ***/
VOID io_XCreatePixmap(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Window parent = (Window) io_host_ioreq_ptr[IO_DATA2].ls;
  auINT32 width = io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 height = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 depth = io_host_ioreq_ptr[IO_DATA5].ls;

 io_write_return_values(
       (auINT32) XCreatePixmap(disp, parent, width, height, depth),0);
}

/*** io_XFreePixmap - free a pixmap ID.
 ***
 *** Assumes:
 ***    IO_DATA1 - disp
 ***    IO_DATA2 - pixmap
 ***/
VOID io_XFreePixmap(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Pixmap pixm = (Pixmap) io_host_ioreq_ptr[IO_DATA2].ls;

  XFreePixmap(disp, pixm);
}

/*** io_XCopyArea - copy an area of a drawable.
 ***
 *** Assumes:
 ***    IO_DATA1 - disp
 ***    IO_DATA2 - src
 ***    IO_DATA3 - dest
 ***    IO_DATA4 - gc
 ***    IO_DATA5 - x
 ***    IO_DATA6 - y
 ***    IO_DATA7 - width
 ***    IO_DATA8 - height
```

```
 ***    IO_DATA9 - dest_x
 ***    IO_DATA10 - dest_y
 ***/
VOID io_XCopyArea(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Drawable src = (Drawable) io_host_ioreq_ptr[IO_DATA2].ls;
  Drawable dest = (Drawable) io_host_ioreq_ptr[IO_DATA3].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 x = io_host_ioreq_ptr[IO_DATA5].ls;
  auINT32 y = io_host_ioreq_ptr[IO_DATA6].ls;
  auINT32 width = io_host_ioreq_ptr[IO_DATA7].ls;
  auINT32 height = io_host_ioreq_ptr[IO_DATA8].ls;
  auINT32 dest_x = io_host_ioreq_ptr[IO_DATA9].ls;
  auINT32 dest_y = io_host_ioreq_ptr[IO_DATA10].ls;

  XCopyArea(disp, src, dest, gc, x, y, width, height, dest_x, dest_y);
}



/*** io_WaitForMap - wait for the next MapEvent.
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***/
VOID io_WaitForMap(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  XEvent event;

  do
    XNextEvent(disp, &event);
  while (event.type != MapNotify);
  io_write_return_values((auINT32) event.xmap.window, 0);
}

/*** io_XDisplayArray - display the content of an array
 ***
 *** Assumes:
 ***    IO_DATA1 - display
 ***    IO_DATA2 - drawable
 ***    IO_DATA3 - gc
 ***    IO_DATA4 - x_win
```

```
 ***    IO_DATA5 - y_win
 ***    IO_DATA6 - width
 ***    IO_DATA7 - height
 ***    IO_DATA8 - nbcol
 ***    IO_DATA9 - alpha
 ***    IO_DATA10 - beta
 ***    IO_BUF   - size of pixel (width & height), vector of color,
                   and array of points (one 64-bits word per point)
 ***/
VOID io_DisplayArray(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  Drawable drawable = (Drawable) io_host_ioreq_ptr[IO_DATA2].ls;
  GC gc = (GC) io_host_ioreq_ptr[IO_DATA3].ls;
  auINT32 x_win = io_host_ioreq_ptr[IO_DATA4].ls;
  auINT32 y_win = io_host_ioreq_ptr[IO_DATA5].ls;
  auINT32 width = io_host_ioreq_ptr[IO_DATA6].ls;
  auINT32 height = io_host_ioreq_ptr[IO_DATA7].ls;
  auINT32 nbcol = io_host_ioreq_ptr[IO_DATA8].ls;
  auINT32 alpha = io_host_ioreq_ptr[IO_DATA9].ls;
  auINT32 beta = io_host_ioreq_ptr[IO_DATA10].ls;

  MPI_INT64 *string_buf, *readpointer;
  auINT32 i, j, col, lastcol;
  auINT32 width_pix, height_pix, x_pix, y_pix;

  string_buf = io_read_from_monsoon_sbuf(
                   io_host_ioreq_ptr,2+nbcol+width*height , IO_SBUF_IN_OH);
  if (alpha==0) alpha=1;
  width_pix = string_buf[0].ls;
  height_pix = string_buf[1].ls;
  lastcol = -1;
  readpointer = &string_buf[2+nbcol];

  if ((width_pix==1)&&(height_pix==1))
    {
      y_pix = y_win;
      for( j=0 ; j<height ; j++)
{
  x_pix = x_win;
  for( i=0 ; i<width ; i++ )
    {
      col=(readpointer->ls+beta)/alpha;
      readpointer++;
```

```
      if (col < 0) col = 0;
      else if (col >= nbcol) col = nbcol-1;
      if (col!=lastcol)
{
  XSetForeground(disp, gc, string_buf[2+col].ls);
  lastcol=col;
}
      XDrawPoint(disp, drawable, gc, x_pix, y_pix);
      x_pix++;
    }
  y_pix++;
}
    }
  else
    {
      y_pix = y_win;
      for( j=0 ; j<height ; j++)
{
  x_pix = x_win;
  for( i=0 ; i<width ; i++ )
    {
      col=(readpointer->ls+beta)/alpha;
      readpointer++;
      if (col < 0) col = 0;
      else if (col >= nbcol) col = nbcol-1;
      if (col!=lastcol)
{
  XSetForeground(disp, gc, string_buf[2+col].ls);
  lastcol=col;
}
      XFillRectangle(disp, drawable, gc,
                             x_pix, y_pix, width_pix, height_pix);
      x_pix += width_pix;
    }
  y_pix += height_pix;
}
    }

  free(string_buf);
}

/*** io_XSynchronize - synchronize the X display.
 ***
 *** Assumes:
```

```
 ***    IO_DATA1 - display
 ***    IO_DATA2 - c_boolean (0 - FALSE, 1 - TRUE)
 ***/
VOID io_XSynchronize(VOID)
{
  Display *disp = (Display *) io_host_ioreq_ptr[IO_DATA1].ls;
  auINT32 bool = io_host_ioreq_ptr[IO_DATA2].ls;

  XSynchronize(disp, bool);
}




/* io_terminate_graphics
 *
 * input:    -
 * output:   -
 * modifies: -
 * effects:  Closes all X display connections in the linked IO_Displays linked
 *           list. A side effect of this is that all windows related to these
 *           display connections are also closed.
 */
VOID io_terminate_graphics(VOID)
{
  io_map_list (IO_Displays, XCloseDisplay);
}

VOID io_invalid_graphics_op(auINT32 op)
{
  fprintf(stderr,
          "\nio_invalid_graphics_op - Invalid operation code: %d\n", op);
}

/*** List utility functions */
/* io_cons
 *
 * input:    car - a pointer to the element to the added to the list.
 *           cdr - a pointer to the a list.
 * output:   a pointer to a list containing the element as its car.
 * modifies: -
 * effects:  Adds the given element to the front of the list.
 */
```

```
io_cons_cell *io_cons(VOID *car, io_cons_cell *cdr)
{
  io_cons_cell *head;

  head = (io_cons_cell *) malloc (sizeof(io_cons_cell));

  head->car = car;
  head->cdr = cdr;

  return head;
}

/* io_map_list
 *
 * input:    list - a list
 *           fn   - a pointer to a function whose type is the same as the
 *                  elements of the list.
 * output:   -
 * modifies: -
 * effects:  Calls the function fn on each element of the list, for side effect
 *           purposes only.
 */
VOID io_map_list (io_cons_cell *list, VOID (*fn)(VOID *arg))
{
  while (list != NULL)
    { fn (list->car);
      list = list->cdr;
    }
}
```

Appendix B

Interface X, côté Id : listing

```
%%% ****************************************************************************
%%% ***                                                                      ***
%%% ***                          Monsoon System                              ***
%%% ***                            X11 Support                               ***
%%% ***                                                                      ***
%%% ***     Designed and implemented  by the members of the Computation      ***
%%% ***     Structures Group of the MIT Laboratory for Computer Science      ***
%%% ***                                                                      ***
%%% *** Copyright (C) 1991   Massachusetts Institute of Technology           ***
%%% ***                       ALL RIGHTS RESERVED                            ***
%%% ***                                                                      ***
%%% ***       No copy of this source code may be made by any means,          ***
%%% ***       electronic or otherwise,  without prior permission of          ***
%%% ***       the Massachusetts Institute of Technology.                     ***
%%% ***                                                                      ***
%%% ****************************************************************************
%%%

%@include "io:code;io-types";
%@include "io:code;io-sys";
@include "io-base";
@include "string";

%%% Alejandro Caro, July, 1991.
%%% Sylvain Huet, May, 1994.

%%%% This file provides routines that call the X11 library on the
%%%% Monsoon host.
```

```
%%%% ************************************************************************
%%%% CONSTANTS
%%% hardcode graphics channel.
defsubst graphics_pe = 0;

%%% **********************************
%%% X Operations

defsubst XOpenDisplay_code {@inline_only} = 100;
defsubst XBlackPixel_code {@inline_only} = 101;
defsubst XWhitePixel_code {@inline_only} = 102;
defsubst XDefaultGC_code {@inline_only} = 103;
defsubst XDefaultRootWindow_code {@inline_only} = 104;
defsubst XDefaultScreen_code {@inline_only} = 105;
defsubst XCloseDisplay_code {@inline_only} = 106;
defsubst XDefaultVisual_code {@inline_only} = 107;
defsubst XFlush_code {@inline_only} = 108;
defsubst XCreateGC_code {@inline_only} = 109;
defsubst XDefaultColormap_code {@inline_only} = 110;
defsubst XDefaultDepth_code {@inline_only} = 111;
defsubst XFreeGC_code {@inline_only} = 112;

defsubst XCreateSimpleWindow_code {@inline_only} = 200;
defsubst XDestroyWindow_code {@inline_only} = 201;
defsubst XMapWindow_code {@inline_only} = 202;
defsubst XSetWindowBackground_code {@inline_only} = 203;
defsubst XUnmapWindow_code {@inline_only} = 204;
defsubst XMoveWindow_code {@inline_only} = 205;
defsubst XResizeWindow_code {@inline_only} = 206;

defsubst XClearArea_code {@inline_only} = 300;
defsubst XClearWindow_code {@inline_only} = 301;
defsubst XDrawPoint_code {@inline_only} = 302;
defsubst XDrawPoints_code {@inline_only} = 303;
defsubst XDrawLine_code {@inline_only} = 304;
defsubst XDrawLines_code {@inline_only} = 305;
defsubst XDrawRectangle_code {@inline_only} = 307;
defsubst XDrawArc_code {@inline_only} = 309;
defsubst XFillRectangle_code {@inline_only} = 311;
defsubst XFillPolygon_code {@inline_only} = 313;
defsubst XFillArc_code {@inline_only} = 314;
defsubst XDrawString_code {@inline_only} = 316;
defsubst XDrawImageString_code {@inline_only} = 317;
```

```
defsubst XSetBackground_code {@inline_only} = 318;
defsubst XSetForeground_code {@inline_only} = 319;
defsubst XSetFunction_code {@inline_only} = 320;
defsubst XSetLineAttributes_code {@inline_only} = 321;

defsubst XSelectInput_code {@inline_only} = 400;
defsubst XNextEvent_code {@inline_only} = 401;
defsubst XCheckTypedEvent_code {@inline_only} = 402;
defsubst XQueryPointer_code {@inline_only} = 403;
defsubst XSendEvent_code {@inline_only} = 404;

defsubst XAllocNamedColor_code {@inline_only} = 500;
defsubst XAllocColor_code {@inline_only} = 501;
defsubst XCreateColormap_code {@inline_only} = 502;
defsubst XSetWindowColormap_code {@inline_only} = 503;
defsubst XFreeColormap_code {@inline_only} = 504;
defsubst XLoadQueryFont_code {@inline_only} = 505;
defsubst XFreeFont_code {@inline_only} = 506;
defsubst XSetFont_code {@inline_only} = 507;

defsubst XCreatePixmap_code {@inline_only} = 600;
defsubst XFreePixmap_code {@inline_only} = 601;
defsubst XCopyArea_code {@inline_only} = 602;

defsubst WaitForMap_code {@inline_only} = 900;
defsubst DisplayArray_code {@inline_only} = 901;

defsubst XSynchronize_code {@inline_only} = 1000;

%%% ***********************************
%%% X Constants

defsubst NoEventMask {@inline_only} = 0;
defsubst KeyPressMask {@inline_only} = 1;
defsubst KeyReleaseMask {@inline_only} = 2;
defsubst ButtonPressMask {@inline_only} = 4;
defsubst ButtonReleaseMask {@inline_only} = 8;
defsubst EnterWindowMask {@inline_only} = 16;
defsubst LeaveWindowMask {@inline_only} = 32;
defsubst ButtonMotionMask {@inline_only} = 8192;
defsubst ExposureMask {@inline_only} = 32768;
defsubst StructureNotifyMask {@inline_only} = 131072;

defsubst NoEvent {@inline_only} = 0;
```

```
defsubst KeyPress {@inline_only} = 2;
defsubst KeyRelease {@inline_only} = 3;
defsubst ButtonPress {@inline_only} = 4;
defsubst ButtonRelease {@inline_only} = 5;
defsubst MotionNotify {@inline_only} = 6;
defsubst EnterNotify {@inline_only} = 7;
defsubst LeaveNotify {@inline_only} = 8;
defsubst Expose {@inline_only} = 12;
defsubst MapNotify {@inline_only} = 19;
defsubst ConfigureNotify {@inline_only} = 22;

defsubst ShiftMask {@inline_only} = 1;
defsubst LockMask {@inline_only} = 2;
defsubst ControlMask {@inline_only} = 4;
defsubst Button1Mask {@inline_only} = 256;
defsubst Button2Mask {@inline_only} = 512;
defsubst Button3Mask {@inline_only} = 1024;

defsubst GXclear {@inline_only} = 0;
defsubst GXand {@inline_only} = 1;
defsubst GXandReverse {@inline_only} = 2;
defsubst GXcopy {@inline_only} = 3;
defsubst GXandInverted {@inline_only} = 4;
defsubst GXnoop {@inline_only} = 5;
defsubst GXxor {@inline_only} = 6;
defsubst GXor {@inline_only} = 7;
defsubst GXnor {@inline_only} = 8;
defsubst GXequiv {@inline_only} = 9;
defsubst GXinvert {@inline_only} = 10;
defsubst GXorReverse {@inline_only} = 11;
defsubst GXcopyInverted {@inline_only} = 12;
defsubst GXorInverted {@inline_only} = 13;
defsubst GXnand {@inline_only} = 14;
defsubst GXset {@inline_only} = 15;

defsubst LineSolid {@inline_only} = 0;
defsubst LineOnOffDash {@inline_only} = 1;
defsubst LineDoubleDash {@inline_only} = 2;

defsubst CapNotLast {@inline_only} = 0;
defsubst CapButt {@inline_only} = 1;
defsubst CapRound {@inline_only} = 2;
defsubst CapProjecting {@inline_only} = 3;
```

```
defsubst JoinMiter {@inline_only} = 0;
defsubst JoinRound {@inline_only} = 1;
defsubst JoinBevel {@inline_only} = 2;

%%%% ********************************************************************
%%%% XEvent Types

type XEvent =  XKeyPress        i i i i i i i i
                              % win x y x_root y_root state keycode ascii
             | XKeyRelease      i i i i i i i i
                              % win x y x_root y_root state keycode ascii
             | XButtonPress     i i i i i i i
                              % win x y x_root y_root state button
             | XButtonRelease   i i i i i i i
                              % win x y x_root y_root state button
             | XMotionNotify    i i i i i i
                              % win x y x_root y_root state
             | XEnterNotify     i i i i i i
                              % win x y x_root y_root state
             | XLeaveNotify     i i i i i i
                              % win x y x_root y_root state
             | XExpose          i i i i i
                              % win x y width height
             | XMapNotify       i
                              % win
             | XConfigureNotify i i i i i
                              % win x y width height
             | XDefault         i ;
                              % type

%%%% ********************************************************************
%%%% DISPLAY FUNCTIONS
%%%%
%%%% To avoid deadlock problems, all the X functions are strict in their
%%%% arguments BEFORE they grab the io_sub_buffer.
def XOpenDisplay display_name =
  { typeof display_name = S;
    ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display_name);
    sbuf,buf_size,trig2 = sys_io_get_sbuf graphics_pe count
                              (to_sys_io_trig display_name);
    sbuf_offset = 0;
    count = length display_name;
    _     = io_write_imp ioreq sys_ioi_class
                    (io_encode_request io_graphics_request);
```

```
    _     = io_write_imp ioreq sys_ioi_opcode XOpenDisplay_code;
    _     = io_write_imp ioreq sys_ioi_sbuf_size count;
    _     = io_write_imp ioreq sys_ioi_sbuf_ptr sbuf;
    _     = io_write_imp ioreq sys_ioi_data1 count;
    t3 = sys_io_fill_sbuf_from_object display_name
                         (io_string_header_offset display_name)
                         sbuf sbuf_offset count trig2;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    display = io_read_imp (gate ioreq rtrig) 0;
   ---
    done1     = sys_io_release_sbuf sbuf buf_size rtrig;
    done2     = sys_io_release_ioreq ioreq rtrig;
  in
    display };


def XBlackPixel display screen_number =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _   = io_write_imp ioreq sys_ioi_class
                         (io_encode_request io_graphics_request);
    _   = io_write_imp ioreq sys_ioi_opcode XBlackPixel_code;
    _   = io_write_imp ioreq sys_ioi_data1 display;
    _   = io_write_imp ioreq sys_ioi_data2 screen_number;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    black_pixel = io_read_imp (gate ioreq rtrig) 0;
   ---
    _     = sys_io_release_ioreq ioreq rtrig;
  in
    black_pixel };


def XWhitePixel display screen_number =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _   = io_write_imp ioreq sys_ioi_class
                         (io_encode_request io_graphics_request);
    _   = io_write_imp ioreq sys_ioi_opcode XWhitePixel_code;
    _   = io_write_imp ioreq sys_ioi_data1 display;
    _   = io_write_imp ioreq sys_ioi_data2 screen_number;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    white_pixel = io_read_imp (gate ioreq rtrig) 0;
    _     = sys_io_release_ioreq ioreq (gate rtrig white_pixel);
```

```
  in
    white_pixel };


def XDefaultGC display screen_number =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _   = io_write_imp ioreq sys_ioi_class
                         (io_encode_request io_graphics_request);
    _   = io_write_imp ioreq sys_ioi_opcode XDefaultGC_code;
    _   = io_write_imp ioreq sys_ioi_data1 display;
    _   = io_write_imp ioreq sys_ioi_data2 screen_number;
   ---
    rtrig   = io_op graphics_pe ioreq trig1;
    default_gc = io_read_imp (gate ioreq rtrig) 0;
    _   = sys_io_release_ioreq ioreq (gate rtrig default_gc);
  in
    default_gc };


def XDefaultRootWindow display =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _   = io_write_imp ioreq sys_ioi_class
                         (io_encode_request io_graphics_request);
    _   = io_write_imp ioreq sys_ioi_opcode XDefaultRootWindow_code;
    _   = io_write_imp ioreq sys_ioi_data1 display;
   ---
    rtrig   = io_op graphics_pe ioreq trig1;
    default_root = io_read_imp (gate ioreq rtrig) 0;
    _   = sys_io_release_ioreq ioreq (gate rtrig default_root);
  in
    default_root };


def XDefaultScreen display =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                          (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XDefaultScreen_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    default_screen_number = io_read_imp (gate ioreq rtrig) 0;
    _     = sys_io_release_ioreq ioreq (gate rtrig default_screen_number);
  in
```

```
    default_screen_number };

def XDefaultVisual display screen =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XDefaultVisual_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 screen;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    default_visual = io_read_imp (gate ioreq rtrig) 0;
    _     = sys_io_release_ioreq ioreq (gate rtrig default_visual);
  in
    default_visual };


def XCloseDisplay display =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XCloseDisplay_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _     = sys_io_release_ioreq ioreq rtrig;
  };

def XFlush display =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XFlush_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _     = sys_io_release_ioreq ioreq rtrig;
  };

def XCreateGC display drawable =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _   = io_write_imp ioreq sys_ioi_class
                       (io_encode_request io_graphics_request);
    _   = io_write_imp ioreq sys_ioi_opcode XCreateGC_code;
```

```
    _   = io_write_imp ioreq sys_ioi_data1 display;
    _   = io_write_imp ioreq sys_ioi_data2 drawable;
   ---
    rtrig   = io_op graphics_pe ioreq trig1;
    new_gc = io_read_imp (gate ioreq rtrig) 0;
    _   = sys_io_release_ioreq ioreq (gate rtrig new_gc);
  in
    new_gc };

def XDefaultColormap display screen_number =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _   = io_write_imp ioreq sys_ioi_class
                       (io_encode_request io_graphics_request);
    _   = io_write_imp ioreq sys_ioi_opcode XDefaultColormap_code;
    _   = io_write_imp ioreq sys_ioi_data1 display;
    _   = io_write_imp ioreq sys_ioi_data2 screen_number;
   ---
    rtrig   = io_op graphics_pe ioreq trig1;
    default_colormap = io_read_imp (gate ioreq rtrig) 0;
    _   = sys_io_release_ioreq ioreq (gate rtrig default_colormap);
  in
    default_colormap };

def XDefaultDepth display screen_number =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _   = io_write_imp ioreq sys_ioi_class
                       (io_encode_request io_graphics_request);
    _   = io_write_imp ioreq sys_ioi_opcode XDefaultDepth_code;
    _   = io_write_imp ioreq sys_ioi_data1 display;
    _   = io_write_imp ioreq sys_ioi_data2 screen_number;
   ---
    rtrig   = io_op graphics_pe ioreq trig1;
    default_depth = io_read_imp (gate ioreq rtrig) 0;
    _   = sys_io_release_ioreq ioreq (gate rtrig default_depth);
  in
    default_depth };

def XFreeGC display gc =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XFreeGC_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 gc;
```

```
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _     = sys_io_release_ioreq ioreq rtrig;
  };

%%%% ********************************************************************
%%%% WINDOW FUNCTIONS

def XCreateSimpleWindow display parent x y width height
    border_width border background =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XCreateSimpleWindow_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 parent;
    _    = io_write_imp ioreq sys_ioi_data3 x;
    _    = io_write_imp ioreq sys_ioi_data4 y;
    _    = io_write_imp ioreq sys_ioi_data5 width;
    _    = io_write_imp ioreq sys_ioi_data6 height;
    _    = io_write_imp ioreq sys_ioi_data7 border_width;
    _    = io_write_imp ioreq sys_ioi_data8 border;
    _    = io_write_imp ioreq sys_ioi_data9 background;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    window = io_read_imp (gate ioreq rtrig) 0;
    _     = sys_io_release_ioreq ioreq (gate rtrig window);
  in
    window };


def XDestroyWindow display window =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _   = io_write_imp ioreq sys_ioi_class
                       (io_encode_request io_graphics_request);
    _   = io_write_imp ioreq sys_ioi_opcode XDestroyWindow_code;
    _   = io_write_imp ioreq sys_ioi_data1 display;
    _   = io_write_imp ioreq sys_ioi_data2 window;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _   = sys_io_release_ioreq ioreq rtrig;
  };
```

```
def XMapWindow display window =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XMapWindow_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 window;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };



def XSetWindowBackground display window background_pixel =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XSetWindowBackground_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 window;
    _    = io_write_imp ioreq sys_ioi_data3 background_pixel;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };

def XUnmapWindow display window =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XUnmapWindow_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 window;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };

def XMoveWindow display drawable x y =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
   ---
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
```

```
    _    = io_write_imp ioreq sys_ioi_opcode XMoveWindow_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 drawable;
    _    = io_write_imp ioreq sys_ioi_data3 x;
    _    = io_write_imp ioreq sys_ioi_data4 y;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };

def XResizeWindow display drawable x y =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
   ---
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XResizeWindow_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 drawable;
    _    = io_write_imp ioreq sys_ioi_data3 x;
    _    = io_write_imp ioreq sys_ioi_data4 y;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };

%%%% ********************************************************************
%%%% GRAPHICS FUNCTIONS

def XClearArea display window x y width height exposures_c_bool =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
   ---
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XClearArea_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 window;
    _    = io_write_imp ioreq sys_ioi_data3 x;
    _    = io_write_imp ioreq sys_ioi_data4 y;
    _    = io_write_imp ioreq sys_ioi_data5 width;
    _    = io_write_imp ioreq sys_ioi_data6 height;
    _    = io_write_imp ioreq sys_ioi_data7 exposures_c_bool;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
```

```
  };


def XClearWindow display window =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
   ---
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XClearWindow_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 window;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };

def XDrawPoint display drawable gc x y =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
   ---
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XDrawPoint_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 drawable;
    _    = io_write_imp ioreq sys_ioi_data3 gc;
    _    = io_write_imp ioreq sys_ioi_data4 x;
    _    = io_write_imp ioreq sys_ioi_data5 y;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };

def XDrawPoints display drawable gc tab mode =
  { (i,j) = bounds~1D_array tab;
    ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    sbuf, buf_size, trig2 = sys_io_get_sbuf graphics_pe count
                                  (to_sys_io_trig display);
    count= (j-i+1)*16;
    sbuf_offset = 0;
   ---
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XDrawPoints_code;
    _    = io_write_imp ioreq sys_ioi_sbuf_size count;
```

```
    _    = io_write_imp ioreq sys_ioi_sbuf_ptr sbuf;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 drawable;
    _    = io_write_imp ioreq sys_ioi_data3 gc;
    _    = io_write_imp ioreq sys_ioi_data4 (j-i+1);
    _    = io_write_imp ioreq sys_ioi_data5 mode;
    _    = {for k <- i to j do
              (x,y) = tab[k];
              _ = sys_io_write_int_to_sbuf sbuf sbuf_offset x trig2;
              _ = sys_io_write_int_to_sbuf sbuf (sbuf_offset+8) y trig2;
              next sbuf_offset = sbuf_offset+16;}
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _   = sys_io_release_ioreq ioreq rtrig;
    _   = sys_io_release_sbuf sbuf buf_size rtrig;
  };

def XDrawLine display drawable gc x1 y1 x2 y2 =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
   ---
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XDrawLine_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 drawable;
    _    = io_write_imp ioreq sys_ioi_data3 gc;
    _    = io_write_imp ioreq sys_ioi_data4 x1;
    _    = io_write_imp ioreq sys_ioi_data5 y1;
    _    = io_write_imp ioreq sys_ioi_data6 x2;
    _    = io_write_imp ioreq sys_ioi_data7 y2;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };

def XDrawLines display drawable gc tab mode =
  { (i,j) = bounds~1D_array tab;
    ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    sbuf, buf_size, trig2 = sys_io_get_sbuf graphics_pe count
                                        (to_sys_io_trig display);
    count= (j-i+1)*16;
    sbuf_offset = 0;
   ---
    _    = io_write_imp ioreq sys_ioi_class
```

```
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XDrawLines_code;
    _    = io_write_imp ioreq sys_ioi_sbuf_size count;
    _    = io_write_imp ioreq sys_ioi_sbuf_ptr sbuf;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 drawable;
    _    = io_write_imp ioreq sys_ioi_data3 gc;
    _    = io_write_imp ioreq sys_ioi_data4 (j-i+1);
    _    = io_write_imp ioreq sys_ioi_data5 mode;
    _    = {for k <- i to j do
              (x,y) = tab[k];
              _ = sys_io_write_int_to_sbuf sbuf sbuf_offset x trig2;
              _ = sys_io_write_int_to_sbuf sbuf (sbuf_offset+8) y trig2;
              next sbuf_offset = sbuf_offset+16;}
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _   = sys_io_release_ioreq ioreq rtrig;
    _   = sys_io_release_sbuf sbuf buf_size rtrig;
  };

def XDrawRectangle display drawable gc x y width height =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
   ---
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XDrawRectangle_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 drawable;
    _    = io_write_imp ioreq sys_ioi_data3 gc;
    _    = io_write_imp ioreq sys_ioi_data4 x;
    _    = io_write_imp ioreq sys_ioi_data5 y;
    _    = io_write_imp ioreq sys_ioi_data6 width;
    _    = io_write_imp ioreq sys_ioi_data7 height;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };

def XDrawArc display drawable gc x y width height angle1 angle2 =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
   ---
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XDrawArc_code;
```

```
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 drawable;
    _    = io_write_imp ioreq sys_ioi_data3 gc;
    _    = io_write_imp ioreq sys_ioi_data4 x;
    _    = io_write_imp ioreq sys_ioi_data5 y;
    _    = io_write_imp ioreq sys_ioi_data6 width;
    _    = io_write_imp ioreq sys_ioi_data7 height;
    _    = io_write_imp ioreq sys_ioi_data8 angle1;
    _    = io_write_imp ioreq sys_ioi_data9 angle2;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };

def XFillRectangle display drawable gc x y width height =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
   ---
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XFillRectangle_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 drawable;
    _    = io_write_imp ioreq sys_ioi_data3 gc;
    _    = io_write_imp ioreq sys_ioi_data4 x;
    _    = io_write_imp ioreq sys_ioi_data5 y;
    _    = io_write_imp ioreq sys_ioi_data6 width;
    _    = io_write_imp ioreq sys_ioi_data7 height;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };

def XFillPolygon display drawable gc tab mode =
  { (i,j) = bounds~1D_array tab;
    ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    sbuf, buf_size, trig2 = sys_io_get_sbuf graphics_pe count
                                        (to_sys_io_trig display);
    count= (j-i+1)*16;
    sbuf_offset = 0;
   ---
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XFillPolygon_code;
    _    = io_write_imp ioreq sys_ioi_sbuf_size count;
```

```
    _    = io_write_imp ioreq sys_ioi_sbuf_ptr sbuf;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 drawable;
    _    = io_write_imp ioreq sys_ioi_data3 gc;
    _    = io_write_imp ioreq sys_ioi_data4 (j-i+1);
    _    = io_write_imp ioreq sys_ioi_data5 mode;
    _    = {for k <- i to j do
              (x,y) = tab[k];
              _ = sys_io_write_int_to_sbuf sbuf sbuf_offset x trig2;
              _ = sys_io_write_int_to_sbuf sbuf (sbuf_offset+8) y trig2;
              next sbuf_offset = sbuf_offset+16;}
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _   = sys_io_release_ioreq ioreq rtrig;
    _   = sys_io_release_sbuf sbuf buf_size rtrig;
  };

def XFillArc display drawable gc x y width height angle1 angle2 =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
   ---
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XFillArc_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 drawable;
    _    = io_write_imp ioreq sys_ioi_data3 gc;
    _    = io_write_imp ioreq sys_ioi_data4 x;
    _    = io_write_imp ioreq sys_ioi_data5 y;
    _    = io_write_imp ioreq sys_ioi_data6 width;
    _    = io_write_imp ioreq sys_ioi_data7 height;
    _    = io_write_imp ioreq sys_ioi_data8 angle1;
    _    = io_write_imp ioreq sys_ioi_data9 angle2;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };

def XDrawString display drawable gc x y string =
  { typeof string = S;
    ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    sbuf, buf_size, trig2 = sys_io_get_sbuf graphics_pe count
                                        (to_sys_io_trig display);
    sbuf_offset = 0;
    count = length string;
```

```
    _   = io_write_imp ioreq sys_ioi_class
                         (io_encode_request io_graphics_request);
    _   = io_write_imp ioreq sys_ioi_opcode XDrawString_code;
    _   = io_write_imp ioreq sys_ioi_sbuf_size count;
    _   = io_write_imp ioreq sys_ioi_sbuf_ptr sbuf;
    _   = io_write_imp ioreq sys_ioi_data1 display;
    _   = io_write_imp ioreq sys_ioi_data2 drawable;
    _   = io_write_imp ioreq sys_ioi_data3 gc;
    _   = io_write_imp ioreq sys_ioi_data4 x;
    _   = io_write_imp ioreq sys_ioi_data5 y;
    _   = io_write_imp ioreq sys_ioi_data6 count;
    trig3 = sys_io_fill_sbuf_from_object string
                       (io_string_header_offset string)
      sbuf sbuf_offset count trig2;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _   = sys_io_release_ioreq ioreq rtrig;
    _   = sys_io_release_sbuf  sbuf buf_size rtrig;
  };

def XDrawImageString display drawable gc x y string =
  { typeof string = S;
    ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    sbuf, buf_size, trig2 = sys_io_get_sbuf graphics_pe count
                                       (to_sys_io_trig display);
    sbuf_offset = 0;
    count = length string;
    _   = io_write_imp ioreq sys_ioi_class
                         (io_encode_request io_graphics_request);
    _   = io_write_imp ioreq sys_ioi_opcode XDrawImageString_code;
    _   = io_write_imp ioreq sys_ioi_sbuf_size count;
    _   = io_write_imp ioreq sys_ioi_sbuf_ptr sbuf;
    _   = io_write_imp ioreq sys_ioi_data1 display;
    _   = io_write_imp ioreq sys_ioi_data2 drawable;
    _   = io_write_imp ioreq sys_ioi_data3 gc;
    _   = io_write_imp ioreq sys_ioi_data4 x;
    _   = io_write_imp ioreq sys_ioi_data5 y;
    _   = io_write_imp ioreq sys_ioi_data6 count;
    trig3 = sys_io_fill_sbuf_from_object string
                       (io_string_header_offset string)
      sbuf sbuf_offset count trig2;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _   = sys_io_release_ioreq ioreq rtrig;
```

```
    _   = sys_io_release_sbuf sbuf buf_size rtrig;
  };

def XSetBackground display gc background_pixel =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XSetBackground_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 gc;
    _    = io_write_imp ioreq sys_ioi_data3 background_pixel;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };

def XSetForeground display gc foreground_pixel =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XSetForeground_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 gc;
    _    = io_write_imp ioreq sys_ioi_data3 foreground_pixel;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };

def XSetFunction display gc function =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XSetFunction_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 gc;
    _    = io_write_imp ioreq sys_ioi_data3 function;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };

def XSetLineAttributes display gc line_width line_style cap_style join_style =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
```

```
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XSetLineAttributes_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 gc;
    _    = io_write_imp ioreq sys_ioi_data3 line_width;
    _    = io_write_imp ioreq sys_ioi_data4 line_style;
    _    = io_write_imp ioreq sys_ioi_data5 cap_style;
    _    = io_write_imp ioreq sys_ioi_data6 join_style;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };


%%%% ************************************************************************
%%%% EVENTS FUNCTIONS

def XSelectInput display window event_mask =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XSelectInput_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 window;
    _    = io_write_imp ioreq sys_ioi_data3 event_mask;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };

def readevent type_event sbuf rtrig =
  {case type_event of
               2 = (XKeyPress (sys_io_read_int_from_sbuf sbuf 0 rtrig)
                              (sys_io_read_int_from_sbuf sbuf 8 rtrig)
                              (sys_io_read_int_from_sbuf sbuf 16 rtrig)
                              (sys_io_read_int_from_sbuf sbuf 24 rtrig)
                              (sys_io_read_int_from_sbuf sbuf 32 rtrig)
                              (sys_io_read_int_from_sbuf sbuf 40 rtrig)
                              (sys_io_read_int_from_sbuf sbuf 48 rtrig)
                              (sys_io_read_int_from_sbuf sbuf 56 rtrig))
             | 3 = (XKeyRelease (sys_io_read_int_from_sbuf sbuf 0 rtrig)
                              (sys_io_read_int_from_sbuf sbuf 8 rtrig)
                              (sys_io_read_int_from_sbuf sbuf 16 rtrig)
```

```
                    (sys_io_read_int_from_sbuf sbuf 24 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 32 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 40 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 48 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 56 rtrig))
  | 4 = (XButtonPress (sys_io_read_int_from_sbuf sbuf 0 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 8 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 16 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 24 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 32 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 40 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 48 rtrig))
  | 5 = (XButtonRelease (sys_io_read_int_from_sbuf sbuf 0 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 8 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 16 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 24 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 32 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 40 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 48 rtrig))
  | 6 = (XMotionNotify (sys_io_read_int_from_sbuf sbuf 0 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 8 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 16 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 24 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 32 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 40 rtrig))
  | 7 = (XEnterNotify (sys_io_read_int_from_sbuf sbuf 0 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 8 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 16 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 24 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 32 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 40 rtrig))
  | 8 = (XLeaveNotify (sys_io_read_int_from_sbuf sbuf 0 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 8 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 16 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 24 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 32 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 40 rtrig))
  | 12 = (XExpose (sys_io_read_int_from_sbuf sbuf 0 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 8 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 16 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 24 rtrig)
                    (sys_io_read_int_from_sbuf sbuf 32 rtrig))
  | 19 = (XMapNotify (sys_io_read_int_from_sbuf sbuf 0 rtrig))
  | 22 = (XConfigureNotify (sys_io_read_int_from_sbuf sbuf 0 rtrig)
```

```
                                (sys_io_read_int_from_sbuf sbuf 8 rtrig)
                                (sys_io_read_int_from_sbuf sbuf 16 rtrig)
                                (sys_io_read_int_from_sbuf sbuf 24 rtrig)
                                (sys_io_read_int_from_sbuf sbuf 32 rtrig))
                |.. n = (XDefault (sys_io_read_int_from_sbuf sbuf 0 rtrig))};

def XNextEvent display =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    sbuf, buf_size, trig2 = sys_io_get_sbuf graphics_pe count
                                        (to_sys_io_trig display);
    count= 10*8;
    ---
    _    = io_write_imp ioreq sys_ioi_class
                          (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XNextEvent_code;
    _    = io_write_imp ioreq sys_ioi_sbuf_size count;
    _    = io_write_imp ioreq sys_ioi_sbuf_ptr sbuf;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    ---
    rtrig = io_op graphics_pe ioreq trig2;
    type_event = io_read_imp (gate ioreq rtrig) 0;
    event = readevent type_event sbuf rtrig;
    ---
    _   = sys_io_release_ioreq ioreq (gate rtrig type_event);
    _   = sys_io_release_sbuf  sbuf buf_size rtrig;
  in
    event };

def XCheckTypedEvent display event_type =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    sbuf, buf_size, trig2 = sys_io_get_sbuf graphics_pe count
                                      (to_sys_io_trig display);
    count= 10*8;
    ---
    _    = io_write_imp ioreq sys_ioi_class
                          (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XCheckTypedEvent_code;
    _    = io_write_imp ioreq sys_ioi_sbuf_size count;
    _    = io_write_imp ioreq sys_ioi_sbuf_ptr sbuf;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 event_type;
    ---
    rtrig = io_op graphics_pe ioreq trig2;
    type_event = io_read_imp (gate ioreq rtrig) 0;
```

```
    event = readevent type_event sbuf rtrig;
    ---
    _   = sys_io_release_ioreq ioreq (gate rtrig type_event);
    _   = sys_io_release_sbuf  sbuf buf_size rtrig;
  in
    event };

def XQueryPointer display win =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    sbuf, buf_size, trig2 = sys_io_get_sbuf graphics_pe count
                                      (to_sys_io_trig display);
    count= 6*8;
    ---
    _    = io_write_imp ioreq sys_ioi_class
                         (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XQueryPointer_code;
    _    = io_write_imp ioreq sys_ioi_sbuf_size count;
    _    = io_write_imp ioreq sys_ioi_sbuf_ptr sbuf;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 win;

    ---
    rtrig = io_op graphics_pe ioreq trig2;
    keys_buttons = io_read_imp (gate ioreq rtrig) 0;
    root = sys_io_read_int_from_sbuf sbuf 0 rtrig;
    child = sys_io_read_int_from_sbuf sbuf 8 rtrig;
    root_x = sys_io_read_int_from_sbuf sbuf 16 rtrig;
    root_y = sys_io_read_int_from_sbuf sbuf 24 rtrig;
    win_x = sys_io_read_int_from_sbuf sbuf 32 rtrig;
    win_y = sys_io_read_int_from_sbuf sbuf 40 rtrig;
    ---
    _   = sys_io_release_ioreq ioreq (gate rtrig keys_buttons);
    _   = sys_io_release_sbuf  sbuf buf_size rtrig;
  in
    (root, child, root_x, root_y, win_x, win_y, keys_buttons) };

def XSendEvent display window event_mask event =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XSendEvent_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 window;
    _    = io_write_imp ioreq sys_ioi_data3 event_mask;
```

```
_    = {case event of
          (XKeyPress win x y x_root y_root state keycode ascii)
                   = { _ = io_write_imp ioreq sys_ioi_data4 2;
                       _ = io_write_imp ioreq sys_ioi_data5 x;
                       _ = io_write_imp ioreq sys_ioi_data6 y;
                       _ = io_write_imp ioreq sys_ioi_data7 x_root;
                       _ = io_write_imp ioreq sys_ioi_data8 y_root;
                       _ = io_write_imp ioreq sys_ioi_data9 state;
                       _ = io_write_imp ioreq sys_ioi_data10 keycode; }
         |(XKeyRelease win x y x_root y_root state keycode ascii)
                   = { _ = io_write_imp ioreq sys_ioi_data4 3;
                       _ = io_write_imp ioreq sys_ioi_data5 x;
                       _ = io_write_imp ioreq sys_ioi_data6 y;
                       _ = io_write_imp ioreq sys_ioi_data7 x_root;
                       _ = io_write_imp ioreq sys_ioi_data8 y_root;
                       _ = io_write_imp ioreq sys_ioi_data9 state;
                       _ = io_write_imp ioreq sys_ioi_data10 keycode; }
         |(XButtonPress win x y x_root y_root state button)
                   = { _ = io_write_imp ioreq sys_ioi_data4 4;
                       _ = io_write_imp ioreq sys_ioi_data5 x;
                       _ = io_write_imp ioreq sys_ioi_data6 y;
                       _ = io_write_imp ioreq sys_ioi_data7 x_root;
                       _ = io_write_imp ioreq sys_ioi_data8 y_root;
                       _ = io_write_imp ioreq sys_ioi_data9 state;
                       _ = io_write_imp ioreq sys_ioi_data10 button; }
         |(XButtonRelease win x y x_root y_root state button)
                   = { _ = io_write_imp ioreq sys_ioi_data4 5;
                       _ = io_write_imp ioreq sys_ioi_data5 x;
                       _ = io_write_imp ioreq sys_ioi_data6 y;
                       _ = io_write_imp ioreq sys_ioi_data7 x_root;
                       _ = io_write_imp ioreq sys_ioi_data8 y_root;
                       _ = io_write_imp ioreq sys_ioi_data9 state;
                       _ = io_write_imp ioreq sys_ioi_data10 button; }
         |(XMotionNotify win x y x_root y_root state)
                   = { _ = io_write_imp ioreq sys_ioi_data4 6;
                       _ = io_write_imp ioreq sys_ioi_data5 x;
                       _ = io_write_imp ioreq sys_ioi_data6 y;
                       _ = io_write_imp ioreq sys_ioi_data7 x_root;
                       _ = io_write_imp ioreq sys_ioi_data8 y_root;
                       _ = io_write_imp ioreq sys_ioi_data9 state; }
         |(XEnterNotify win x y x_root y_root state)
                   = { _ = io_write_imp ioreq sys_ioi_data4 7;
                       _ = io_write_imp ioreq sys_ioi_data5 x;
                       _ = io_write_imp ioreq sys_ioi_data6 y;
```

```
                              _ = io_write_imp ioreq sys_ioi_data7 x_root;
                              _ = io_write_imp ioreq sys_ioi_data8 y_root;
                              _ = io_write_imp ioreq sys_ioi_data9 state; }
                |(XLeaveNotify win x y x_root y_root state)
                          = { _ = io_write_imp ioreq sys_ioi_data4 8;
                              _ = io_write_imp ioreq sys_ioi_data5 x;
                              _ = io_write_imp ioreq sys_ioi_data6 y;
                              _ = io_write_imp ioreq sys_ioi_data7 x_root;
                              _ = io_write_imp ioreq sys_ioi_data8 y_root;
                              _ = io_write_imp ioreq sys_ioi_data9 state; }
                |(XExpose win x y width height)
                          = { _ = io_write_imp ioreq sys_ioi_data4 12;
                              _ = io_write_imp ioreq sys_ioi_data5 x;
                              _ = io_write_imp ioreq sys_ioi_data6 y;
                              _ = io_write_imp ioreq sys_ioi_data7 width;
                              _ = io_write_imp ioreq sys_ioi_data8 height; }
                |(XMapNotify win)
                          = { _ = io_write_imp ioreq sys_ioi_data4 19; }
                |(XConfigureNotify win x y width height)
                          = { _ = io_write_imp ioreq sys_ioi_data4 22;
                              _ = io_write_imp ioreq sys_ioi_data5 x;
                              _ = io_write_imp ioreq sys_ioi_data6 y;
                              _ = io_write_imp ioreq sys_ioi_data7 width;
                              _ = io_write_imp ioreq sys_ioi_data8 height; }
                |.. _     = { _ = io_write_imp ioreq sys_ioi_data4 0; }
                };
    ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };

%%%% ************************************************************************
%%%% COLOR FUNCTIONS

def XAllocNamedColor display colormap string =
  { typeof string = S;
    ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    sbuf, buf_size, trig2 = sys_io_get_sbuf graphics_pe count
                                        (to_sys_io_trig display);
    sbuf_offset = 0;
    count = length string;
    _   = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _   = io_write_imp ioreq sys_ioi_opcode XAllocNamedColor_code;
```

```
    _   = io_write_imp ioreq sys_ioi_sbuf_size count;
    _   = io_write_imp ioreq sys_ioi_sbuf_ptr sbuf;
    _   = io_write_imp ioreq sys_ioi_data1 display;
    _   = io_write_imp ioreq sys_ioi_data2 colormap;
    _   = io_write_imp ioreq sys_ioi_data3 count;
    trig3 = sys_io_fill_sbuf_from_object string
                      (io_string_header_offset string)
      sbuf sbuf_offset count trig2;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    color_code = io_read_imp (gate ioreq rtrig) 0;
    _   = sys_io_release_ioreq ioreq (gate rtrig color_code);
    _   = sys_io_release_sbuf  sbuf buf_size (gate rtrig color_code);
  in color_code };

def XAllocColor display colormap red green blue =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _   = io_write_imp ioreq sys_ioi_class
                       (io_encode_request io_graphics_request);
    _   = io_write_imp ioreq sys_ioi_opcode XAllocColor_code;
    _   = io_write_imp ioreq sys_ioi_data1 display;
    _   = io_write_imp ioreq sys_ioi_data2 colormap;
    _   = io_write_imp ioreq sys_ioi_data3 red;
    _   = io_write_imp ioreq sys_ioi_data4 green;
    _   = io_write_imp ioreq sys_ioi_data5 blue;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    color_code = io_read_imp (gate ioreq rtrig) 0;
    _   = sys_io_release_ioreq ioreq (gate rtrig color_code);
  in color_code };

def XCreateColormap display window visual =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XCreateColormap_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 window;
    _    = io_write_imp ioreq sys_ioi_data3 visual;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    new_colormap = io_read_imp (gate ioreq rtrig) 0;
    _     = sys_io_release_ioreq ioreq (gate rtrig new_colormap);
  in
```

```
    new_colormap };

def XSetWindowColormap display window cmap =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XSetWindowColormap_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 window;
    _    = io_write_imp ioreq sys_ioi_data3 cmap;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _     = sys_io_release_ioreq ioreq rtrig;
  };

def XFreeColormap display cmap =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XFreeColormap_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 cmap;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _     = sys_io_release_ioreq ioreq rtrig;
  };

def XLoadQueryFont display string =
  { typeof string = S;
    ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    sbuf, buf_size, trig2 = sys_io_get_sbuf graphics_pe count
                                        (to_sys_io_trig display);
    sbuf_offset = 0;
    count = length string;
    _   = io_write_imp ioreq sys_ioi_class
                       (io_encode_request io_graphics_request);
    _   = io_write_imp ioreq sys_ioi_opcode XLoadQueryFont_code;
    _   = io_write_imp ioreq sys_ioi_sbuf_size count;
    _   = io_write_imp ioreq sys_ioi_sbuf_ptr sbuf;
    _   = io_write_imp ioreq sys_ioi_data1 display;
    _   = io_write_imp ioreq sys_ioi_data2 count;
    trig3 = sys_io_fill_sbuf_from_object string
                       (io_string_header_offset string)
      sbuf sbuf_offset count trig2;
```

```
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    font_info = io_read_imp (gate ioreq rtrig) 0;
    _   = sys_io_release_ioreq ioreq (gate rtrig font_info);
    _   = sys_io_release_sbuf  sbuf buf_size (gate rtrig font_info);
  in font_info };

def XFreeFont display font =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XFreeFont_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 font;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _     = sys_io_release_ioreq ioreq rtrig;
  };

def XSetFont display gc font =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XSetFont_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 gc;
    _    = io_write_imp ioreq sys_ioi_data3 font;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _     = sys_io_release_ioreq ioreq rtrig;
  };


%%%% ************************************************************************
%%%% PIXMAP FUNCTIONS

def XCreatePixmap display window width height depth =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XCreatePixmap_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 window;
    _    = io_write_imp ioreq sys_ioi_data3 width;
```

```
    _    = io_write_imp ioreq sys_ioi_data4 height;
    _    = io_write_imp ioreq sys_ioi_data5 depth;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    new_pixmap = io_read_imp (gate ioreq rtrig) 0;
    _     = sys_io_release_ioreq ioreq (gate rtrig new_pixmap);
  in
    new_pixmap };

def XFreePixmap display pixmap =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XFreePixmap_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 pixmap;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _     = sys_io_release_ioreq ioreq rtrig;
  };

def XCopyArea display src dest gc x y width height dest_x dest_y =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XCopyArea_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 src;
    _    = io_write_imp ioreq sys_ioi_data3 dest;
    _    = io_write_imp ioreq sys_ioi_data4 gc;
    _    = io_write_imp ioreq sys_ioi_data5 x;
    _    = io_write_imp ioreq sys_ioi_data6 y;
    _    = io_write_imp ioreq sys_ioi_data7 width;
    _    = io_write_imp ioreq sys_ioi_data8 height;
    _    = io_write_imp ioreq sys_ioi_data9 dest_x;
    _    = io_write_imp ioreq sys_ioi_data10 dest_y;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _     = sys_io_release_ioreq ioreq rtrig;
  };


%%%% ************************************************************************
%%%% USEFUL FUNCTIONS
```

```
def WaitForMap display =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _   = io_write_imp ioreq sys_ioi_class
                       (io_encode_request io_graphics_request);
    _   = io_write_imp ioreq sys_ioi_opcode WaitForMap_code;
    _   = io_write_imp ioreq sys_ioi_data1 display;
   ---
    rtrig   = io_op graphics_pe ioreq trig1;
    mapped_window = io_read_imp (gate ioreq rtrig) 0;
    _   = sys_io_release_ioreq ioreq (gate rtrig mapped_window);
  in  mapped_window };

def DisplayArray display drawable gc x_win y_win tab colors
                                  alpha beta width_pix height_pix=
  { (i_col,j_col) = bounds~1D_array colors;
    ((i_x,j_x),(i_y,j_y)) = bounds~2D_array tab;
    ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    sbuf, buf_size, trig2 = sys_io_get_sbuf graphics_pe count
                                       (to_sys_io_trig display);
    count= (2+(j_col-i_col+1)+(j_x-i_x+1)*(j_y-i_y+1))*8;
    sbuf_offcol = 16;
    sbuf_offtaby = (j_col - i_col+1)*8+16;
   ---
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode DisplayArray_code;
    _    = io_write_imp ioreq sys_ioi_sbuf_size count;
    _    = io_write_imp ioreq sys_ioi_sbuf_ptr sbuf;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 drawable;
    _    = io_write_imp ioreq sys_ioi_data3 gc;
    _    = io_write_imp ioreq sys_ioi_data4 x_win;
    _    = io_write_imp ioreq sys_ioi_data5 y_win;
    _    = io_write_imp ioreq sys_ioi_data6 (j_x-i_x+1);
    _    = io_write_imp ioreq sys_ioi_data7 (j_y-i_y+1);
    _    = io_write_imp ioreq sys_ioi_data8 (j_col-i_col+1);
    _    = io_write_imp ioreq sys_ioi_data9 alpha;
    _    = io_write_imp ioreq sys_ioi_data10 beta;
    _    = sys_io_write_int_to_sbuf sbuf 0 width_pix trig2;
    _    = sys_io_write_int_to_sbuf sbuf 8 height_pix trig2;
    _    = {for k <- i_col to j_col do
              _ = sys_io_write_int_to_sbuf sbuf sbuf_offcol colors[k] trig2;
              next sbuf_offcol = sbuf_offcol+8;};
```

```
    _    = {for y <- i_y to j_y do
              sbuf_offtabx = sbuf_offtaby;
              next sbuf_offtaby =
                {for x <- i_x to j_x do
                   _ = sys_io_write_int_to_sbuf
                                sbuf sbuf_offtabx tab[x,y] trig2;
                   next sbuf_offtabx = sbuf_offtabx+8;
                finally sbuf_offtabx}
           }
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _   = sys_io_release_ioreq ioreq rtrig;
    _   = sys_io_release_sbuf sbuf buf_size rtrig;
  };

%%%% ********************************************************************
%%%% MISCELLANEOUS FUNCTIONS

def XSynchronize display c_bool =
  { ioreq, trig1 = sys_io_get_ioreq graphics_pe (to_sys_io_trig display);
    _    = io_write_imp ioreq sys_ioi_class
                        (io_encode_request io_graphics_request);
    _    = io_write_imp ioreq sys_ioi_opcode XSynchronize_code;
    _    = io_write_imp ioreq sys_ioi_data1 display;
    _    = io_write_imp ioreq sys_ioi_data2 c_bool;
   ---
    rtrig = io_op graphics_pe ioreq trig1;
    _    = sys_io_release_ioreq ioreq rtrig;
  };
```

Appendix C

Breakout : listing

```
@include "/jj/sylvain/proj/graph/iographics";

def initableau disp win gc =
{ _ = { for i <- 0 to 9 do
         { for j <- 0 to 5 do
            _ = XFillRectangle disp win gc (i*40+2) (j*20+42) 36 16;
         };
      };
  _ = XFillRectangle disp win gc 160 370 80 10;
  ---
  _ = XFlush disp;
};

def newdir x =
  if (x<35) then (if (x<5) then (-10,-5)
                  else if (x<20) then (-10,-10)
                       else (-5,-10))
            else (if (x<50) then (5,-10)
                  else if (x<65) then (10,-10)
                       else (10,-5));

def afterchoc dx dy x y =
{ yi = if (dy<0) then 20 else 0;
  xp = x+ (quo (dx * (yi - y)) dy);
  in if ((xp<0)or(xp>40)) then (-dx,dy) else (dx,-dy)};

def newbutton disp win =
{ (_,_,_,_,x,y,butt) = XQueryPointer disp win;
 in if ((x<0)or(x>400)or(y<0)or(y>400)) then 0 else butt
```

```
};

def boucleev disp win gc =
{ button = 0;
  xj = 160;
  xb = 195;
  yb = 195;
  dx = 10;
  dy = -5;
  m = {M_matrix ((0,9),(0,5)) of [i,j]=1 || i<- 0 to 9 ; j <- 0 to 5};
  _ = XFillRectangle disp win gc xb yb 10 10;
  ---
  _ = {while (button<>Button2Mask) do
         butt = newbutton disp win;
         nx = if ((butt==Button1Mask)and(xj>0)) then
              { _ = XFillRectangle disp win gc (xj-10) 370 10 10;
                _ = XFillRectangle disp win gc (xj+70) 370 10 10;
               ---
               in (xj-10)}
         else if ((butt==Button3Mask)and(xj<320)) then
              { _ = XFillRectangle disp win gc xj 370 10 10;
                _ = XFillRectangle disp win gc (xj+80) 370 10 10;
               ---
               in (xj+10)}
         else xj;
         _ = XFillRectangle disp win gc xb yb 10 10;
         (nxb, ndx) = if (xb+dx>390) then (780-xb-dx,-dx)
                      else if (xb+dx<0) then (-xb-dx,-dx)
                      else ((xb+dx),dx);
         (nyb, ndy) = if (yb+dy>390) then (780-yb-dy,-dy)
                      else if (yb+dy<0) then (-yb-dy,-dy)
                      else ((yb+dy),dy);
         (mdx, mdy) = if ((nyb<360)or(nyb-ndy>360)
                          or(nxb<nx-10)or(nxb>nx+80))
                       then (ndx,ndy)
                      else newdir (nxb-nx);
         xcent= quo (nxb+5) 40;
         ycent= (quo (nyb+5) 20)-2;
         (pdx, pdy) = if ((ycent<0)or(ycent>5)) then (mdx,mdy)
                 else if (m![xcent,ycent]==0) then (mdx,mdy)
                 else { _ = XFillRectangle disp win gc
                                (xcent*40+2) (ycent*20+42) 36 16;
                        m![xcent,ycent] = 0;
                       ---
```

```
                        in afterchoc mdx mdy (mod (nxb+5) 40)
                                             (mod (nyb+5) 20)};
          _ = XFillRectangle disp win gc nxb nyb 10 10;
          next button = butt;
          next xj = nx;
          next xb = nxb;
          next yb = nyb;
          next dx = pdx;
          next dy = pdy;
          ---
      };
 ---
 in ()};

def breakout name =
{ disp = XOpenDisplay name;
  root = XDefaultRootWindow disp;
  scrn = XDefaultScreen disp;
  whi = XWhitePixel disp scrn;
  bla = XBlackPixel disp scrn;
  gc = XDefaultGC disp scrn;
  _ = XSetForeground disp gc (logxor bla whi);
  win = XCreateSimpleWindow disp root 100 100 400 400 1 whi bla;
  _ = XSelectInput disp win (StructureNotifyMask);
  _ = XSetFunction disp gc GXxor;
  ---
  _ = XMapWindow disp win;
  ---
  _ = WaitForMap disp;
  ---
  _ = initableau disp win gc;
  ---
  _ = boucleev disp win gc;
  ---
  _ = XUnmapWindow disp win;
  ---
  _ = XDestroyWindow disp win;
  ---
  in XFlush disp};
```

Appendix D

Jeu de la Vie : listing

```
@include "/jj/sylvain/proj/graph/iographics";

def inicolor cw cb = {array (0,1) of [i]=(if (i==0) then cb else cw)
                                    || i <- 0 to 1};

def inimtab x y = {M_matrix ((1,x),(1,y)) of [i,j] = 0
                                    || i <- 1 to x; j <- 1 to y};

def copymtab m =
{
((xmin,xmax),(ymin,ymax)) = bounds~2D_M_array m;
res = {matrix ((xmin,xmax),(ymin,ymax)) of [i,j] = m!![i,j]
                      || i <- xmin to xmax; j <- ymin to ymax};
in res
};

def createtab disp win gc col m =
{ ((xmin,xmax),(ymin,ymax)) = bounds~2D_M_array m;
  (colmin,colmax) = bounds~1D_array col;
  bigloop = 2;
  _ = {while (bigloop==2) do
         _ = {for i <- xmin to xmax do
                {for j <- ymin to ymax do
                   m!![i,j] = 0}};
         _ = XSetForeground disp gc col[0];
         ---
         _ = XFillRectangle disp win gc 0 0
                        ((xmax-xmin+1)*16) ((ymax-ymin+1)*16);
```

```
          ---
          _ = XSetForeground disp gc col[1];
          ---
          _ = {for i<- xmin to (xmax-1) do
                 colon = (i-xmin)*16+15;
                 maxli = (ymax-ymin+1)*16;
                 _ = XDrawLine disp win gc colon 0 colon maxli;
              };
          _ = {for j<- ymin to (ymax-1) do
                 ligne = (j-ymin)*16+15;
                 maxco = (xmax-xmin+1)*16;
                 _ = XDrawLine disp win gc 0 ligne maxco ligne;
              };
          ---
          _ = XFlush disp;
          ---
          contin = 1;
          next bigloop =
             { while (contin==1) do
                 ev = XNextEvent disp;
                 next contin = {case ev of
                 (XButtonPress w x y xr yr st but) =
                        if (but==1) then
                          { xc = quo x 16;
                            yc = quo y 16;
                            xt = xmin + xc;
                            yt = ymin + yc;
                            newcol = 1 - m![xt, yt];
                            m![xt, yt] = newcol;
                            _ = XSetForeground disp gc col[newcol];
                            ---
                            _ = XFillRectangle disp win gc
                                        (xc*16+1) (yc*16+1) 13 13;
                             in 1}
                        else but
                 |.. _ = 1}
                ---
                 finally contin};
              };
  ---
  in (copymtab m)
};

def newstate tab i j =
```

```
{ ((xmin,xmax),(ymin,ymax)) = bounds~2D_array tab;
  hau = if (j>ymin) then tab[i, j-1] else 0;
  bas = if (j<ymax) then tab[i, j+1] else 0;
  gau = if (i>xmin) then tab[i-1, j] else 0;
  dro = if (i<xmax) then tab[i+1, j] else 0;
  hga = if ((i>xmin)and(j>ymin)) then tab[i-1, j-1] else 0;
  hdr = if ((i<xmax)and(j>ymin)) then tab[i+1, j-1] else 0;
  bga = if ((i>xmin)and(j<ymax)) then tab[i-1, j+1] else 0;
  bdr = if ((i<xmax)and(j<ymax)) then tab[i+1, j+1] else 0;

  stat = hau + bas + gau + dro + hga + hdr + bga + bdr;
  actua = tab[i, j];
  in
  if ((stat==3 - actua)or(stat==3)) then 1 else 0
};

def runlife disp win gc col tab =
{ ((xmin,xmax),(ymin,ymax)) = bounds~2D_array tab;
  quit = 0;
  cod = {while (quit==0) do
         @release tab;
         ev = XCheckTypedEvent disp ButtonPress;
         next quit =  if ((XButtonPress? ev)==true) then
                        { (XButtonPress _ _ _ _ _ _ but) = ev;
                          in but}
                        else 0;
         next tab = {matrix ((xmin,xmax),(ymin,ymax)) of
                             [i,j] = (newstate tab i j)
                           || i <- xmin to xmax; j <- ymin to ymax};
         _ = DisplayArray disp win gc 0 0 tab col 1 0 16 16;
         ---
         finally quit};
 ---
 in cod};

def life name taille =
{ disp = XOpenDisplay name;
  root = XDefaultRootWindow disp;
  scrn = XDefaultScreen disp;
  whi = XWhitePixel disp scrn;
  bla = XBlackPixel disp scrn;
  col = inicolor whi bla;
  gc = XDefaultGC disp scrn;
  _ = XSetForeground disp gc whi;
```

```
tailwin = taille*16;
win = XCreateSimpleWindow disp root 200 200 tailwin tailwin 1 whi bla;
_ = XSelectInput disp win (StructureNotifyMask+ButtonPressMask);
---
_ = XMapWindow disp win;
---
_ = WaitForMap disp;
---
exit = 0;
_ = { while (exit<>3) do
       m = inimtab taille taille;
     ---
       tab = createtab disp win gc col m ;
     ---
       next exit = runlife disp win gc col tab ;
     };
---
_ = XUnmapWindow disp win;
---
_ = XDestroyWindow disp win;
---
in XFlush disp};
```

Appendix E

Excel : listing

E.1 Analyseur syntaxique

```
@include "string";

type noeud =
    val i
  | add noeud noeud
  | sub noeud noeud
  | mul noeud noeud
  | dvd noeud noeud
  | cell i i
  | noth;

def is_a_number a =
  { asc = (char_to_int a) - 48;
    in
      if ((asc<0)or (asc>9)) then (false,0)
      else (true, asc)};

def is_a_char a =
  { asc = (char_to_int a) - 65;
    in
      if ((asc>=0)and (asc<26)) then (true,asc)
      else if ((asc>=32) and (asc<32+26)) then (true, (asc-32))
      else (false, 0)};

def get_seq  nil  n f = if (f) then (n, nil) else ((-1), nil)
|   get_seq (a:l) n f =
```

```
  { (flag, ch) = is_a_number a;
   in
     if (flag) then get_seq l (n*10+ch) true
     else if (f) then (n, (a:l)) else ((-1), nil)};

def get_number nil = ((noth), nil)
|   get_number (a:l) =
   { p = if (a=='-') then l else (a:l);
     (n, q) = get_seq p 0 false;
    in
      if (n<0) then ((noth), (a:l))
      else if (a=='-') then ((val (-n)), q)
      else ((val n), q)};

def get_alp  nil  n f = if (f) then (n, nil) else ((-1), nil)
|   get_alp (a:l) n f =
  { (flag, ch) = is_a_char a;
   in
     if (flag) then get_alp l ((n+1)*26+ch) true
     else if (f) then (n, (a:l)) else ((-1), nil)};

def get_cel nil = ((noth),nil)
|   get_cel (a:l) =
   { (n, p) = get_alp (a:l) (-1) false;
     (m, q) = if (n<0) then (n, p) else get_seq p 0 false;
    in
      if (m<0) then ((noth), (a:l)) else ((cell n m), q)};

def get_term nil = ((noth), nil)
|   get_term (a:l) =
  if (a=='(') then
    { (expr, p) = get_expr l;
     in if (noth? expr) then ((noth),(a:l))
        else {case p of
                nil = ((noth),(a:l))
              | (b:q) = if (b==')') then (expr, q)
                        else ((noth),(a:l))
             }
     }
  else
    { (numb, p) = get_number (a:l);
     in
       if (noth? numb) then
         { (cel, p) = get_cel (a:l);
```

```
          in
            if (noth? cel) then ((noth),(a:l))
            else (cel, p)
          }
        else (numb, p)
    };

def get_prod l =
{ (gauch, p) = get_term l;
   contin = true;
 in
   if (noth? gauch) then ((noth),l)
   else {while contin do
           (next contin, next gauch, next p)=
             { case p of
                 nil  = (false, gauch, p)
               | (a:q) = if ((a=='*')or(a=='/')) then
                         { (droit, r) = get_term q;
                          in
                            if (noth? droit) then (false,(noth),l)
                            else if (a=='*') then (true, (mul gauch droit), r)
                            else (true, (dvd gauch droit), r)
                         }
                       else (false, gauch, p)
             }
           finally (gauch, p)}
};

def get_expr l =
{ (gauch, p) = get_prod l;
   contin = true;
 in
   if (noth? gauch) then ((noth),l)
   else {while contin do
           (next contin, next gauch, next p)=
             { case p of
                 nil  = (false, gauch, p)
               | (a:q) = if ((a=='+')or(a=='-')) then
                         { (droit, r) = get_prod q;
                          in
                            if (noth? droit) then (false,(noth),l)
                            else if (a=='+') then (true, (add gauch droit), r)
                            else (true, (sub gauch droit), r)
                         }
```

```
                    else (false, gauch, p)
             }
           finally (gauch, p)}
};
```

E.2 Fonctions de base

```
%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Basic Functions

@include "string";
@include "/jj/sylvain/proj/excel/syntax";


def is_empty nil = true
|   is_empty (a:l) = false;

%%%%%%%%%%%%%%%%
% int_to_string
%

def pos_to_str n s = if (n==0) then s else pos_to_str (quo n 10)
                                        ((int_to_char ((mod n 10)+48)):s);

def int_to_str n =
{ mylist =
    if (n==0) then ('0':nil)
    else if (n<0) then ('-':(pos_to_str (-n) nil)) else pos_to_str n nil;
 in
   list_to_string mylist};


%%%%%%%%%%%%%%%%
% list of char -> string of length n, with special format
%

def ext_str_to_n s n =
{ len = length~string s;
```

```
 in
   if (len<n) then (ext_str_to_n (conc~string s " ") n) else s
};

def lst_to_promp l n =
{ s = conc~string(conc~string "]" (list_to_string l)) "<";
in ext_str_to_n s n };

%%%%%%%%%%%%%%%%
% edit_the_list : add one char or backspace
%

def lst_backspace nil = nil
|   lst_backspace (a:nil) = nil
|   lst_backspace (a:b:l) = (a:(lst_backspace (b:l)));

def lst_add_char_bis nil c i n = if (i<n) then (c:nil) else nil
|   lst_add_char_bis (a:l) c i n = (a:(lst_add_char_bis l c (i+1) n));

def lst_add_char l c n = lst_add_char_bis l c 0 n;

def lst_add_lst l nil n = l
|   lst_add_lst l (a:p) n = lst_add_lst (lst_add_char l a n) p n;

%%%%%%%%%%%%%%%%
% cell_to_string : give a string that represents a cell
%

def format_str s =
{ len = length~string s;
 in if (len>5) then "  Big"
               else if (len<5) then (format_str (conc~string " " s))
                               else s
};

def lst_lett x l =
{ res = (quo x 26) -1;
  ch = int_to_char (65 + (mod x 26));
 in
   if (res<0) then (ch:l)
   else lst_lett res (ch:l)
};

def int_to_lett x = list_to_string (lst_lett x nil);
```

```
def cell_to_str (val n) i = int_to_str n
|   cell_to_str (add n1 n2) i =
              { s = conc~string (conc~string (cell_to_str n1 0) "+")
                                              (cell_to_str n2 0);
                in
                  if (i>0) then conc~string (conc~string "(" s) ")"
                  else s }
|   cell_to_str (sub n1 n2) i =
              { s = conc~string (conc~string (cell_to_str n1 0) "-")
                                              (cell_to_str n2 1);
                in
                  if (i>0) then conc~string (conc~string "(" s) ")"
                  else s }
|   cell_to_str (mul n1 n2) i =
              { s = conc~string (conc~string (cell_to_str n1 1) "*")
                                              (cell_to_str n2 1);
                in
                  if (i>1) then conc~string (conc~string "(" s) ")"
                  else s }
|   cell_to_str (dvd n1 n2) i =
              { s = conc~string (conc~string (cell_to_str n1 1) "/")
                                              (cell_to_str n2 2);
                in
                  if (i>1) then conc~string (conc~string "(" s) ")"
                  else s }
|   cell_to_str (cell x y) i = conc~string (int_to_lett x) (int_to_str y)
|   cell_to_str (noth) i = "";

def res_to_str (val n) sh = format_str (int_to_str n)
|   res_to_str (noth)  sh = if (noth? sh) then "     "
                            else "Undef";


def cut_list_5 nil n = if (n<5) then (' ':(cut_list_5 nil (n+1))) else nil
|   cut_list_5 (a:l) n = if ((n==4)and((is_empty l)==false)) then
                                                ((int_to_char 92):nil)
                         else if (n<5) then (a:(cut_list_5 l (n+1))) else nil;

def cell_to_shortstr c = list_to_string (cut_list_5 (string_to_list
                                                    (cell_to_str c 0)) 0);

%%%%%%%%%%%%%%%
% operations on ordered lists of (x,y)
```

```
%

typeof add_lst_ord = (i,i) -> (list (i,i)) -> (list (i,i));
typeof del_lst_ord = (i,i) -> (list (i,i)) -> (list (i,i));
typeof in_lst_ord = (i,i) -> (list (i,i)) -> b;
typeof fus_lst_ord = (list (i,i)) -> (list (i,i)) -> (list (i,i));
typeof in_lst_desord = (i,i) -> (list (i,i)) -> b;


def add_lst_ord (x,y) nil = ((x,y):nil)
|   add_lst_ord (x,y) ((a,b):l) = if ((x<a)or ((x==a)and(y<b)))
                                    then ((x,y):(a,b):l)
                                    else if ((x==a)and(y==b)) then ((a,b):l)
                                    else ((a,b):(add_lst_ord (x,y) l));

def del_lst_ord (x,y) nil = (nil)
|   del_lst_ord (x,y) ((a,b):l) = if ((x<a)or ((x==a)and(y<b))) then ((a,b):l)
                                    else if ((x==a)and(y==b)) then l
                                    else ((a,b):(del_lst_ord (x,y) l));

def in_lst_ord (x,y) nil = false
|   in_lst_ord (x,y) ((a,b):l) = if ((x<a)or ((x==a)and(y<b))) then false
                                    else if ((x==a)and(y==b)) then true
                                    else (in_lst_ord (x,y) l);

def fus_lst_ord   nil        l      = l
|   fus_lst_ord ((x,y):l)   nil     = ((x,y):l)
|   fus_lst_ord ((x,y):p) ((a,b):q) = if ((x<a)or ((x==a)and(y<b)))
                                      then ((x,y):(fus_lst_ord p ((a,b):q)))
                                      else if ((x==a)and(y==b))
                                      then ((x,y):(fus_lst_ord p q))
                                      else ((a,b):(fus_lst_ord ((x,y):p) q));

def in_lst_desord (x,y) nil = false
|   in_lst_desord (x,y) ((a,b):l) = if ((x==a)and(y==b)) then true
                                      else (in_lst_desord (x,y) l);

%%%%%%%%%%%%%%%%
% operations on lstdp : list of dependances
%

def free_expr_lst lstdp i j (val n) = 0
|   free_expr_lst lstdp i j (add n1 n2) = (free_expr_lst lstdp i j n1)+
                                          (free_expr_lst lstdp i j n2)
```

```
|   free_expr_lst lstdp i j (sub n1 n2) = (free_expr_lst lstdp i j n1)+
                                          (free_expr_lst lstdp i j n2)
|   free_expr_lst lstdp i j (mul n1 n2) = (free_expr_lst lstdp i j n1)+
                                          (free_expr_lst lstdp i j n2)
|   free_expr_lst lstdp i j (dvd n1 n2) = (free_expr_lst lstdp i j n1)+
                                          (free_expr_lst lstdp i j n2)
|   free_expr_lst lstdp i j (noth) = 0
|   free_expr_lst lstdp i j (cell a b) =
  { ((xn,xx),(yn,yx)) = bounds~2D_M_array lstdp;
    in if ((a<xn)or(a>xx)or(b<yn)or(b>yx)) then 0
       else
         { p = lstdp![a,b];
           q = del_lst_ord (i,j) p;
           lstdp![a,b] = q;
          in 1 }
  };

def free_cell_lstdp sheet lstdp i j =
{ ((_,_),(_,_)) = bounds~2D_M_array sheet;
  expr = sheet![i,j];
  n = free_expr_lst lstdp i j expr
 in n };

def aff_expr_lst lstdp i j (val n) = 0
|   aff_expr_lst lstdp i j (add n1 n2) = (aff_expr_lst lstdp i j n1)+
                                          (aff_expr_lst lstdp i j n2)
|   aff_expr_lst lstdp i j (sub n1 n2) = (aff_expr_lst lstdp i j n1)+
                                          (aff_expr_lst lstdp i j n2)
|   aff_expr_lst lstdp i j (mul n1 n2) = (aff_expr_lst lstdp i j n1)+
                                          (aff_expr_lst lstdp i j n2)
|   aff_expr_lst lstdp i j (dvd n1 n2) = (aff_expr_lst lstdp i j n1)+
                                          (aff_expr_lst lstdp i j n2)
|   aff_expr_lst lstdp i j (noth) = 0
|   aff_expr_lst lstdp i j (cell a b) =
  { ((xn,xx),(yn,yx)) = bounds~2D_M_array lstdp;
    in if ((a<xn)or(a>xx)or(b<yn)or(b>yx)) then 0
       else
         { p = lstdp![a,b];
           q = add_lst_ord (i,j) p;
           lstdp![a,b] = q;
          in 1 }
  };

def aff_cell_lstdp sheet lstdp i j cl =
```

```
{ ((_,_),(_,_)) = bounds~2D_M_array sheet;
  n = aff_expr_lst lstdp i j cl;
  sheet![i,j] = cl;
 in n };
```

E.3 Evaluation

```
@include "/jj/sylvain/proj/graph/iographics";
@include "/jj/sylvain/proj/excel/syntax";
@include "/jj/sylvain/proj/excel/excelbase";

def wholesheet i j =
{ (ni, nj) = if (i<9) then ((i+1), j) else (0, (j+1));
 in
   if (nj==10) then ((i,j):nil) else ((i,j):(wholesheet ni nj))
};

def ref_sheet disp win gc0 gc1 res sheet nil = 0
|   ref_sheet disp win gc0 gc1 res sheet ((x,y):l) =
  { ((_,_),(_,_)) = bounds~2D_M_array res;
    ((_,_),(_,_)) = bounds~2D_M_array sheet;
    c = res!![x,y];
    sh = sheet!![x,y];
    s = res_to_str c sh;
    xs = x*50 + 3;
    ys = y*40 + 58 + 17;
    gc = if (noth? sh) then gc1 else gc0;
    _ = XDrawImageString disp win gc xs ys s;
   in
     (1 + (ref_sheet disp win gc0 gc1 res sheet l)) };

def add_cell (val a) (val b) = (val (a+b))
|.. add_cell x y = (noth);

def sub_cell (val a) (val b) = (val (a-b))
|.. sub_cell x y = (noth);

def mul_cell (val a) (val b) = (val (a*b))
|.. mul_cell x y = (noth);

def dvd_cell (val a) (val b) = if (b==0) then (noth) else (val (quo a b))
|.. dvd_cell x y = (noth);
```

```
def eval_cell res (val n) = (val n)
|   eval_cell res (add n1 n2) = (add_cell (eval_cell res n1)
                                          (eval_cell res n2))
|   eval_cell res (sub n1 n2) = (sub_cell (eval_cell res n1)
                                          (eval_cell res n2))
|   eval_cell res (mul n1 n2) = (mul_cell (eval_cell res n1)
                                          (eval_cell res n2))
|   eval_cell res (dvd n1 n2) = (dvd_cell (eval_cell res n1)
                                          (eval_cell res n2))
|   eval_cell res (noth) = (noth)
|   eval_cell res (cell a b) =
        { ((xn,xx),(yn,yx)) = bounds~2D_M_array res;
         in
           if ((a<xn)or(a>xx)or(b<yn)or(b>yx)) then (noth) else res!![a,b] };

def evallst sheet res nil = 0
|   evallst sheet res ((i,j):l) =
{ ((_,_),(_,_)) = bounds~2D_M_array sheet;
  ((_,_),(_,_)) = bounds~2D_M_array res;
  c = sheet!![i,j];
  res![i,j] = eval_cell res c;
  n = 1 + evallst sheet res l;
 in
    n};

def freelst res nil = 0
|   freelst res ((i,j):l) =
{ ((_,_),(_,_)) = bounds~2D_M_array res;
  _ = res![i,j];
  n = 1 + freelst res l;
 in
   n};

def computelst sheet res l =
{ _ = freelst res l;
 ---
  n = evallst sheet res l;
 in
   n};


def sonlst lstdp nil version newvers = (nil,0)
|   sonlst lstdp ((a,b):l) version newvers =
```

```
  { (p,n) = globlst lstdp a b version newvers;
    (q,m) = sonlst lstdp l version newvers;
   in
     ((p++q),(m+n))};


def globlst lstdp i j version newvers =
{ ((_,_),(_,_)) = bounds~2D_M_array lstdp;
  ((_,_),(_,_)) = bounds~2D_M_array version;

  lastver = version![i,j];
 ---
  version![i,j] = newvers;
 ---
in  if (eq~int lastver newvers) then (nil,0)
    else
      { family = lstdp!![i,j];
        (l,n) = sonlst lstdp family version newvers;
       in
         (((i,j):l),(n+1))}
};

def checkloop l (val n) = true
|   checkloop l (add n1 n2) = ((checkloop l n1)and(checkloop l n2))
|   checkloop l (sub n1 n2) = ((checkloop l n1)and(checkloop l n2))
|   checkloop l (mul n1 n2) = ((checkloop l n1)and(checkloop l n2))
|   checkloop l (dvd n1 n2) = ((checkloop l n1)and(checkloop l n2))
|   checkloop l (cell a b) = if (in_lst_desord (a,b) l) then false else true
|   checkloop l (noth) = true;
```

E.4 Boucle d'événements

```
@include "/jj/sylvain/proj/graph/iographics";
@include "/jj/sylvain/proj/excel/syntax";
@include "/jj/sylvain/proj/excel/excelbase";
@include "/jj/sylvain/proj/excel/exceleval";
@include "string";

defsubst x_val {@inline_only} = 20;
defsubst y_val {@inline_only} = 20;
defsubst x_end {@inline_only} = 450;
defsubst y_end {@inline_only} = 20;
```

```
defsubst maxsz {@inline_only} = 20;

def excelloop disp win gc gci width height =
{ dx = width - 1;
  dy = height - 1;
  lstin = nil;
  contin = true ;
  sheet = {M_matrix ((0,dx),(0,dy)) of [i,j] = (noth)
                                  || i <- 0 to dx; j <- 0 to dy};
  res = {M_matrix ((0,dx),(0,dy)) of [i,j] = (noth)
                                || i <- 0 to dx; j <- 0 to dy};
  lstdp = {M_matrix ((0,dx),(0,dy)) of [i,j] = (nil)
                                  || i <- 0 to dx; j <- 0 to dy};
  version = {M_matrix ((0,dx),(0,dy)) of [i,j] = 0
                                    || i <- 0 to dx; j <- 0 to dy};
  maxvers = 100;
  newvers = 1;
 ---
  _ = { while (contin) do
         ev = XNextEvent disp;
         (next contin, next lstin, vers) = {case ev of
          (XKeyPress w x y xr yr st key asc) =
                   if ((asc>31)and(asc<128)) then
                     { l = lst_add_char lstin (int_to_char asc) maxsz;
                       s = lst_to_promp l (maxsz+2);
                       _ = XDrawImageString disp win gc x_val y_val s;
                      in
                        (true,l,newvers)}
                   else if (asc==8) then
                     { l = lst_backspace lstin;
                       s = lst_to_promp l (maxsz+2);
                       _ = XDrawImageString disp win gc x_val y_val s;
                      in
                        (true,l,newvers)}
                   else (true,lstin,newvers)

         |(XButtonPress w x y xr yr st but) =
                   if (y>40) then
                     (if (but==1) then
                        { xc = quo x 50;
                          yc = quo (y-40) 40;
                          _ = free_cell_lstdp sheet lstdp xc yc;
                % remove sheet[xc,yc], and set up the lists of dependance
                          ---
```

```
                    (lsons, nl) = globlst lstdp xc yc version newvers;
                    (c1,q) = get_expr lstin;
                    ok = checkloop lsons c1;
                    (c,l) = if ((ok)and(is_empty q)) then (c1,nil)
                                                     else ((noth),lstin);
                    _ = aff_cell_lstdp sheet lstdp xc yc c;
          % set sheet[xc,yc], and set up the lists of dependance
                    xs = xc*50 + 3;
                    ys = yc*40 + 58;
                    s = cell_to_shortstr c;
                    _ = XDrawImageString disp win gc xs ys s;
                   ---
                    _ = computelst sheet res lsons;
                   ---
                    _ = ref_sheet disp win gci gc res sheet lsons;
                    sn = format_str (int_to_str nl);
                    _ = XDrawImageString disp win gc 370 20 sn;
                    s2 = lst_to_promp l (maxsz+2);
                    _ = XDrawImageString disp win gc x_val y_val s2;
                   in
                    (true,l,(newvers+1))}
                else if (but==2) then
                       { xc = quo x 50;
                         yc = quo (y-40) 40;
                         c = sheet!![xc,yc];
                         s = cell_to_str c 0;
                         l = string_to_list s;
                         s2 = lst_to_promp l (maxsz+2);
                         _ = XDrawImageString disp win gc x_val y_val s2;
                        in
                          (true,l,newvers)}
                 else { xc = quo x 50;
                        yc = quo (y-40) 40;
                        l = string_to_list (conc~string
                                     (int_to_lett xc) (int_to_str yc));
                        nl = lst_add_lst lstin l maxsz;
                        s2 = lst_to_promp nl (maxsz+2);
                        _ = XDrawImageString disp win gc x_val y_val s2;
                       in
                         (true,nl,newvers)})
     else  if (x>x_end) then (false,lstin,newvers)
     else (true,lstin,newvers)
|.. _ =  (true,lstin,newvers)};
next newvers = if (vers>maxvers) then
```

```
                          { _ = { for j <- 0 to dy do
                                  { for i <- 0 to dx do
                                     version!![i,j] = 0
                                  }
                                };
                            in 1 }
                        else vers;
      ---};
};
```

E.5 Programme principal

```
@include "/jj/sylvain/proj/graph/iographics";
@include "/jj/sylvain/proj/excel/excelbase";
@include "/jj/sylvain/proj/excel/excelloop";

def inigraph disp win gc gci width height =
{ sx = 50*width;
  sy = 40*height+40;
  _ = {for i <- 1 to (width-1) do
         _ = XDrawLine disp win gc (i*50) 40 (i*50) sy;
      };
  _ = {for j <- 1 to height do
         _ = XDrawLine disp win gc 0 (j*40) sx (j*40);
      };
  _ = XDrawLine disp win gc 0 30 sx 30;
  _ = XDrawImageString disp win gc x_end y_end "Quit";
  _ = XDrawImageString disp win gc x_val y_val "]<";
};

def excel name width height =
{ disp = XOpenDisplay name;
  root = XDefaultRootWindow disp;
  scrn = XDefaultScreen disp;
  whi = XWhitePixel disp scrn;
  bla = XBlackPixel disp scrn;

  gc = XDefaultGC disp scrn;
  _ = XSetForeground disp gc whi;
  _ = XSetBackground disp gc bla;

  gci = XCreateGC disp root;
```

```
_ = XSetForeground disp gci bla;
_ = XSetBackground disp gci whi;

font = XLoadQueryFont disp "9x15";
_ = XSetFont disp gc font;
_ = XSetFont disp gci font;

win = XCreateSimpleWindow disp root 200 200
                    (width*50) (40+height*40) 1 whi bla;
_ = XSelectInput disp win (StructureNotifyMask+ButtonPressMask+KeyPressMask);
---
_ = XMapWindow disp win;
---
_ = WaitForMap disp;
---
_ = inigraph disp win gc gci width height;
---
_ = excelloop disp win gc gci width height;
---
_ = XUnmapWindow disp win;
---
_ = XDestroyWindow disp win;
---
in XFlush disp};
```

Appendix F

Problème des n corps : listing

F.1 Fonctions de découpage

```
type Tbody = { record m = F ;
                      x = F ;
                      y = F ;
                      z = F ;
                      vx = F ;
                      vy = F ;
                      vz = F ;
             };

typeof setboundbis = (array Tbody) -> I -> I -> (F, F, F, F, F, F) ;
typeof SetBound = (array Tbody) -> (F, F, F, F) ;

typeof Cut_x = (array Tbody) -> (array I) -> I -> I -> F -> ((array I),I) ;
typeof Cut_y = (array Tbody) -> (array I) -> I -> I -> F -> ((array I),I) ;
typeof Cut_z = (array Tbody) -> (array I) -> I -> I -> F -> ((array I),I) ;

typeof Cut_d = (array Tbody) -> (array I) -> I -> I -> F -> (F, F, F)
             -> ((array I),I) ;

typeof MIN? = F -> F -> F ;
typeof MAX? = F -> F -> F ;

def MIN? a b = if (a<b) then a else b ;
```

```
def MAX? a b = if (a>b) then a else b ;

% compute boundary of whole space

def setboundbis nbody_tab min max =
  if (min<max) then
    { midd = quo (min+max) 2 ;
      (xn1, xx1, yn1, yx1, zn1, zx1) = setboundbis nbody_tab min midd ;
      (xn2, xx2, yn2, yx2, zn2, zx2) = setboundbis nbody_tab (midd+1) max ;
     in
       ((MIN? xn1 xn2),(MAX? xx1 xx2),(MIN? yn1 yn2)
       ,(MAX? yx1 yx2),(MIN? zn1 zn2),(MAX? zx1 zx2))
    }
  else
    { x = nbody_tab[min].x ;
      y = nbody_tab[min].y ;
      z = nbody_tab[min].z ;
     in
       (x,x,y,y,z,z)
    };

def SetBound nbody_tab =
{ (min, max) = bounds nbody_tab ;
  (xn, xx, yn, yx, zn, zx) = setboundbis nbody_tab min max ;
  len = MAX? (MAX? (xx-xn) (yx-yn)) (zx-zn) ;
 in
   (xn, yn, zn, len)
};


% divide an array of points by the plane X = x

def Cut_x nbody_tab tab_index min max x =
  if (min<max) then
    { midd = quo (min+max) 2 ;
      (p, a) = Cut_x nbody_tab tab_index min midd x ;
      (q, b) = Cut_x nbody_tab tab_index (midd+1) max x ;
      (_, ax) = bounds p ;
      (_, bx) = bounds q ;
      s = a + b ;
      t = ax + b ;
      u = a - s ;
      v = ax + 1 ;
      sz = max - min ;
```

```
      ( m = {array (0, sz) of
             | [i] = p[i] || i <- 0 to (a-1)
             | [i] = q[i-a] || i <- a to (s-1)
             | [i] = p[i+u] || i <- s to t
             | [i] = q[i-v] || i <- (t+1) to sz
            };
       ---
        @ release p;
        @ release q;       )
     in
       (m, s)
    }
  else
    { k = tab_index[min] ;
      xp = nbody_tab[k].x;
      ind = if (xp > x) then 0 else 1 ;
      m = {array (0,0) of k} ;
     in
       (m, ind)
    };

% divide an array of points by the plane Y = y

def Cut_y nbody_tab tab_index min max y =
  if (min<max) then
    { midd = quo (min+max) 2 ;
      (p, a) = Cut_y nbody_tab tab_index min midd y ;
      (q, b) = Cut_y nbody_tab tab_index (midd+1) max y ;
      (_, ax) = bounds p ;
      (_, bx) = bounds q ;
      s = a + b ;
      t = ax + b ;
      u = a - s ;
      v = ax + 1 ;
      sz = max - min ;
      ( m = {array (0, sz) of
             | [i] = p[i] || i <- 0 to (a-1)
             | [i] = q[i-a] || i <- a to (s-1)
             | [i] = p[i+u] || i <- s to t
             | [i] = q[i-v] || i <- (t+1) to sz
            };
       ---
        @ release p;
        @ release q;       )
```

```
     in
       (m, s)
    }
  else
    { k = tab_index[min] ;
      yp = nbody_tab[k].y;
      ind = if (yp > y) then 0 else 1 ;
      m = {array (0,0) of k} ;
     in
       (m, ind)
    };


% divide an array of points by the plane Z = z

def Cut_z nbody_tab tab_index min max z =
  if (min<max) then
    { midd = quo (min+max) 2 ;
      (p, a) = Cut_z nbody_tab tab_index min midd z ;
      (q, b) = Cut_z nbody_tab tab_index (midd+1) max z ;
      (_, ax) = bounds p ;
      (_, bx) = bounds q ;
      s = a + b ;
      t = ax + b ;
      u = a - s ;
      v = ax + 1 ;
      sz = max - min ;
      ( m = {array (0, sz) of
             | [i] = p[i] || i <- 0 to (a-1)
             | [i] = q[i-a] || i <- a to (s-1)
             | [i] = p[i+u] || i <- s to t
             | [i] = q[i-v] || i <- (t+1) to sz
            };
       ---
        @ release p;
        @ release q;       )
     in
       (m, s)
    }
  else
    { k = tab_index[min] ;
      zp = nbody_tab[k].z;
      ind = if (zp > z) then 0 else 1 ;
      m = {array (0,0) of k} ;
```

```
     in
       (m, ind)
    };

% divide an array of points by the sphere ((x,y,z),di) where
%       di is the square radius

def Cut_d nbody_tab tab_index min max di (x,y,z) =
  if (min<max) then
    { midd = quo (min+max) 2 ;
      (p, a) = Cut_d nbody_tab tab_index min midd di (x,y,z) ;
      (q, b) = Cut_d nbody_tab tab_index (midd+1) max di (x,y,z) ;
      (_, ax) = bounds p ;
      (_, bx) = bounds q ;
      s = a + b ;
      t = ax + b ;
      u = a - s ;
      v = ax + 1 ;
      sz = max - min ;
      ( m = {array (0, sz)of
             | [i] = p[i] || i <- 0 to (a-1)
             | [i] = q[i-a] || i <- a to (s-1)
             | [i] = p[i+u] || i <- s to t
             | [i] = q[i-v] || i <- (t+1) to sz
            };
       ---
        @ release p;
        @ release q;       )
     in
       (m, s)
    }
  else
    { k = tab_index[min] ;
      dx = nbody_tab[k].x - x ;
      dy = nbody_tab[k].y - y ;
      dz = nbody_tab[k].z - z ;
      dist = dx*dx + dy*dy + dz*dz ;
      ind = if (dist < di) then 1 else 0 ;
      m = {array (0,0) of k} ;
     in
       (m, ind)
    };
```

F.2 Calcul des interactions

```
@include "/jj/sylvain/proj/nbody/bodbase";

defsubst THETA = 0.5 ;
defsubst DISTMIN = 0.0025 ;

type Tree = null
          | onebody i
          | subx Tree Tree f f f f f % sz xc yc zc mc
          | suby Tree Tree
          | subz Tree Tree;



def maketreebis nbody_tab tab_index min max x y z sz =
  if (min>max) then (null)
  else if (min==max) then (onebody (tab_index[min]))
  else { nsz = sz / 2.0 ;
         nx = x + nsz ;

         (m,ind) = Cut_x nbody_tab tab_index min max nx ;
         (_, tot) = bounds m ;
         p = secondlevel nbody_tab m 0 (ind-1) x y z nsz ;
         q = secondlevel nbody_tab m ind tot nx y z nsz ;
        ---
         @release m;

         (l1, x1, y1, z1, m1) = p;
         (l2, x2, y2, z2, m2) = q;

         mc = m1 + m2 ;
         xc = (m1*x1 + m2*x2) / mc ;
         yc = (m1*y1 + m2*y2) / mc ;
         zc = (m1*z1 + m2*z2) / mc ;
        in
          (subx l1 l2 sz xc yc zc mc)
       };

def secondlevel nbody_tab tab_index min max nx y z nsz =
  if (min>max) then ((null), 0.0, 0.0, 0.0, 0.0)
  else { ny = y + nsz ;

         (m,ind) = Cut_y nbody_tab tab_index min max ny ;
```

```
         (_, tot) = bounds m ;
         p = thirdlevel nbody_tab m 0 (ind-1) nx y z nsz ;
         q = thirdlevel nbody_tab m ind tot nx ny z nsz ;
        ---
         @release m;

         (l1, x1, y1, z1, m1) = p;
         (l2, x2, y2, z2, m2) = q;

         mc = m1 + m2 ;
         xc = (m1*x1 + m2*x2) / mc ;
         yc = (m1*y1 + m2*y2) / mc ;
         zc = (m1*z1 + m2*z2) / mc ;
        in
          ((suby l1 l2), xc, yc, zc, mc)
       };


def thirdlevel nbody_tab tab_index min max nx ny z nsz =
  if (min>max) then ((null), 0.0, 0.0, 0.0, 0.0)
  else { nz = z + nsz ;

         (m,ind) = Cut_z nbody_tab tab_index min max nz ;
         (_, tot) = bounds m ;
         p = fourthlevel nbody_tab m 0 (ind-1) nx ny z nsz ;
         q = fourthlevel nbody_tab m ind tot nx ny nz nsz ;
        ---
         @release m;

         (l1, x1, y1, z1, m1) = p;
         (l2, x2, y2, z2, m2) = q;

         mc = m1 + m2 ;
         xc = (m1*x1 + m2*x2) / mc ;
         yc = (m1*y1 + m2*y2) / mc ;
         zc = (m1*z1 + m2*z2) / mc ;
        in
          ((subz l1 l2), xc, yc, zc, mc)
       };

def fourthlevel nbody_tab tab_index min max nx ny nz nsz =
{ res = maketreebis nbody_tab tab_index min max nx ny nz nsz;
 in
   { case res of
```

```
        (null) = (res, 0.0, 0.0, 0.0, 0.0)
       |(onebody i) = (res, nbody_tab[i].x, nbody_tab[i].y, nbody_tab[i].z
                      , nbody_tab[i].m)
       |(subx _ _ _ x y z m) = (res, x, y, z, m)
   }
};


def MakeTree nbody_tab =
{ (_, NBODY) = bounds nbody_tab ;
  (x, y, z, sz) = SetBound nbody_tab ;
  tab_index = {array (0, NBODY) of [i] = i || i <- 0 to NBODY};
  BH_tree = maketreebis nbody_tab tab_index 0 NBODY x y z sz ;
 ---
  @release tab_index ;
 in
   BH_tree
};


typeof addacc = (array Tbody) -> (array I) -> (M_array (F,F,F)) -> I -> I
              -> F -> F -> F -> F -> I ;

def addacc nbody_tab tab_index acc mn mx xc yc zc mc =
{ n = mx - mn + 1 ;
  b = max 2 n ;
  _ = { for i <- mn to mx bound b do
         j = tab_index[i] ;
         dx = xc - nbody_tab[j].x ;
         dy = yc - nbody_tab[j].y ;
         dz = zc - nbody_tab[j].z ;
         dist = dx*dx + dy*dy + dz*dz ;
         (ax, ay, az) = if (dist< DISTMIN) then (0.0, 0.0, 0.0)
                        else { fact = mc / dist / (sqrt dist) ;
                               in
                                 ((fact*dx), (fact*dy), (fact*dz))
                             };
       ---
         (lx, ly, lz) = acc![j] ;
       ---
         acc![j] = ((lx+ax), (ly+ay), (lz+az)) ;
      }
 in
   n
```

```
};


def computeaccbis nbody_tab tab_index acc min max (null) = 0
|   computeaccbis nbody_tab tab_index acc min max (onebody i) =
  { xc = nbody_tab[i].x ;
    yc = nbody_tab[i].y ;
    zc = nbody_tab[i].z ;
    mc = nbody_tab[i].m ;
   in
     addacc nbody_tab tab_index acc min max xc yc zc mc
  }
|   computeaccbis nbody_tab tab_index acc min max (subx p q sz xc yc zc mc) =
  { d = sz / THETA ;
    (t, i) = Cut_d nbody_tab tab_index min max (d*d) (xc, yc, zc) ;
    (_,tot) = bounds t ;
    n1 = if (i>0) then ( (computeaccbis nbody_tab t acc 0 (i-1) p)
                       + (computeaccbis nbody_tab t acc 0 (i-1) q) )
         else 0 ;
    n2 = if (i>tot) then 0
         else addacc nbody_tab t acc i tot xc yc zc mc ;
   ---
    @release t ;
   in
     (n1 + n2)
  }
|   computeaccbis nbody_tab tab_index acc min max (suby p q) =
       ( (computeaccbis nbody_tab tab_index acc min max p)
       + (computeaccbis nbody_tab tab_index acc min max q) )
|   computeaccbis nbody_tab tab_index acc min max (subz p q) =
       ( (computeaccbis nbody_tab tab_index acc min max p)
       + (computeaccbis nbody_tab tab_index acc min max q) );

def ComputeAcceleration nbody_tab tree =
{ (_, NBODY) = bounds nbody_tab ;
  acc = {M_array (0, NBODY) of [i] = (0.0, 0.0, 0.0) || i <- 0 to NBODY};
  tab_index = {array (0, NBODY) of [i] = i || i <- 0 to NBODY};
  n = computeaccbis nbody_tab tab_index acc 0 NBODY tree ;
 ---
  @release tab_index ;
 in
   (acc,n)
};
```

```
def nextpos lastp acc dt =
{ (ax, ay, az) = acc ;
 in
   { record
       m = lastp.m ;
       x = lastp.x + dt*lastp.vx ;
       y = lastp.y + dt*lastp.vy ;
       z = lastp.z + dt*lastp.vz ;
       vx = lastp.vx + dt*ax ;
       vy = lastp.vy + dt*ay ;
       vz = lastp.vz + dt*az ;
   }
};

def MoveBodies nbody_tab acc dt =
{ (_, NBODY) = bounds~1D_array nbody_tab ;
 in
   {array (0, NBODY) of [i] = (nextpos nbody_tab[i] acc![i] dt)
                || i <- 0 to NBODY }
};

def Release_tree tree =
{case tree of
    (null) = ()
   |(onebody i) = {@release tree}
   |(subx p q _ _ _ _ _) = { _ = Release_tree p ;
                            _ = Release_tree q ;
                           ---
                            @release tree}
   |(suby p q) = { _ = Release_tree p ;
                   _ = Release_tree q ;
                  ---
                   @release tree}
   |(subz p q) = { _ = Release_tree p ;
                   _ = Release_tree q ;
                  ---
                   @release tree}
};

def NextStep nbody_tab dt =
{ tree = MakeTree nbody_tab ;
  (acc,n) = ComputeAcceleration nbody_tab tree ;
 ---
  newbody_tab = MoveBodies nbody_tab acc dt ;
```

```
  _ = Release_tree tree ;
 ---
  @release acc ;
 in
   (newbody_tab,n)
};
```

F.3 Interface graphique

```
@include "/jj/sylvain/proj/graph/iographics" ;
@include "/jj/sylvain/proj/nbody/bodbase" ;
@include "/jj/sylvain/proj/nbody/bodloop" ;
@include "/jj/sylvain/proj/nbody/inigalax" ;
@include "string" ;

def pos_to_str n s = if (n==0) then s else pos_to_str (quo n 10)
                                      ((int_to_char ((mod n 10)+48)):s);

def int_to_str n =
{ mylist =
    if (n==0) then ('0':nil)
    else if (n<0) then ('-':(pos_to_str (-n) nil)) else pos_to_str n nil;
 in
   list_to_string mylist};

def GetXY x y z sx sy (aa,ab,ac,ba,bb,bc,ca,cb,cc) (dx,dy,dz) d =
{ x1 = aa*x + ab*y + ac*z + dx ;
  y1 = ba*x + bb*y + bc*z + dy ;
  z1 = ca*x + cb*y + cc*z + dz ;
  (xp,yp) = if (((float sx)*z1 > d*x1) and ((float sx)*z1 > (-d)*x1)
                and ((float sy)*z1 > d*y1) and ((float sy)*z1 > (-d)*y1))
             then ( (sx + truncate (d*x1/z1))
                  , (sy - truncate (d*y1/z1)) )
            else ((-1),(-1)) ;
 in
   (xp,yp)
};

def Makemat th ph rh =
{ sp = sin ph ;
  cp = cos ph ;
  st = sin th ;
  ct = cos th ;
```

```
 in
  ( ((-sp), 0.0, cp, (-st)*cp, ct, (-st)*sp, (-ct)*cp, (-st), (-ct)*sp)
  , (0.0, 0.0, rh) )
};


def drawpoints disp win gc nbody_tab sx sy th ph rh =
{ (_,NBODY) = bounds nbody_tab ;
  (mat, vec) = Makemat th ph rh ;
  points = { array (0,NBODY) of [i] =
               (GetXY nbody_tab[i].x nbody_tab[i].y nbody_tab[i].z
                                                    sx sy mat vec 200.0)
             || i <- 0 to NBODY };
  _ = XDrawPoints disp win gc points 0 ;
  sizelin = rh / 5.0 ;
  (xx, yx) = GetXY sizelin 0.0 0.0 sx sy mat vec 200.0 ;
  (xy, yy) = GetXY 0.0 sizelin 0.0 sx sy mat vec 200.0 ;
  (xz, yz) = GetXY 0.0 0.0 sizelin sx sy mat vec 200.0 ;
  _ = XDrawLine disp win gc sx sy xx yx ;
  _ = XDrawLine disp win gc sx sy xy yy ;
  _ = XDrawLine disp win gc sx sy xz yz ;
 ---
  @release points ;
 in ()
};

def depph ph dx = ph + (float dx) / 100.0 ;

def depth th dx =
{ newth = th + (float dx) / 100.0 ;
  lim = pi / 2.0 ;
 in
   if (newth > lim) then lim
        else if (newth < (-lim)) then (-lim)
        else newth
};

def deprh rh dx =
{ rap = if (dx > (-100)) then (1.0 + (float dx) / 100.0)
        else 0.01 ;
 in
   (rh*rap)
};
```

```
def depobs dx dy th ph rh st =
  if (st==Button1Mask) then ((depth th (-dy)), (depph ph dx), rh)
  else (th, (depph ph dx), (deprh rh dy)) ;

def newbutt disp win sx sy =
{ (_,_,_,_,x,y,butt) = XQueryPointer disp win ;
 in
   if ((x<0)or(x>sx)or(y<0)or(y>sy)) then 0
   else butt
};

def boucle disp win gc gcb nbody_tab dt sx sy =
{ contin = true ;
  (th, ph, rh) = (0.0, 0.0, 10.0) ;
  (xm, ym) = (0, 0) ;
  _ = {while (contin) do
         ev = XNextEvent disp;
         (next contin, next th, next ph, next rh
         , next nbody_tab, next xm, next ym)
            = {case ev of
                  (XExpose w x y dx dy) =
                     { (newbody_tab, n) = NextStep nbody_tab dt ;
                       _ = drawpoints disp win gc nbody_tab sx sy th ph rh ;
                       _ = XDrawImageString disp win gc 5 20 (int_to_str n) ;
                      ---
                       @release nbody_tab ;
                       _ = XSendEvent disp win ExposureMask
                                      (XExpose win 0 0 sx sy) ;
                      in
                        (true, th, ph, rh, newbody_tab, xm, ym)
                     }
                | (XButtonPress _ _ _ x y _ but) =
                     if (but==3) then (false, th, ph, rh, nbody_tab, x, y)
                     else (true, th, ph, rh, nbody_tab, x, y)
                | (XMotionNotify _ _ _ x y st) =
                     { again = true ;
                       (xn, yn) = {while (again) do
                                     nev = XCheckTypedEvent disp MotionNotify ;
                                     (next again, next x, next y) =
                                       { case nev of
                                          (XMotionNotify _ _ _ x1 y1 st) =
                                                 (true, x1, y1)
                                         |.. _ = (false, x, y)
                                       };
```

```
                                    ---
                                      finally (x,y)
                                    };
                       (nth, nph, nrh) = depobs (xn-xm) (yn-ym) th ph rh st ;
                      ---
                       _ = XFillRectangle disp win gcb 0 0 (sx+sx) (sy+sy) ;
                      ---
                       _ = drawpoints disp win gc nbody_tab sx sy nth nph nrh ;
                      in
                        (true, nth, nph, nrh, nbody_tab, xn, yn)
                     }
               |.. _ = (true, th, ph, rh, nbody_tab, xm, ym)
             }
       ---
     }
 in
   ()
};

def test3d name param dt =
{ sx = 200 ;
  sy = 200 ;
  disp = XOpenDisplay name ;
  root = XDefaultRootWindow disp ;
  scrn = XDefaultScreen disp ;
  whi = XWhitePixel disp scrn ;
  bla = XBlackPixel disp scrn ;
  gc = XDefaultGC disp scrn ;
  _ = XSetBackground disp gc bla ;
  _ = XSetForeground disp gc whi ;
  gcb = XCreateGC disp root ;
  _ = XSetBackground disp gcb whi ;
  _ = XSetForeground disp gcb bla ;
  win = XCreateSimpleWindow disp root 100 100 (sx+sx) (sy+sy) 1 whi bla ;
  _ = XSelectInput disp win (StructureNotifyMask+ExposureMask+ButtonPressMask
                            +ButtonMotionMask) ;
 ---
  _ = XMapWindow disp win ;
 ---
  _ = WaitForMap disp ;
 ---
  _ = XSendEvent disp win ExposureMask (XExpose win 0 0 sx sy) ;
  (n,nbody_tab) = initbodies param ;
  _ = XDrawImageString disp win gc 5 40 (int_to_str n) ;
```

```
 ---
  _ = boucle disp win gc gcb nbody_tab dt sx sy;
 ---
  _ = XUnmapWindow disp win ;
 ---
  _ = XDestroyWindow disp win ;
 ---
 in
    XFlush disp
};
```

F.4 Exemple d'initialisation aléatoire

```
@include "/jj/sylvain/proj/nbody/bodbase" ;

def newrand x = sin (x * 100.0) ;

def directunit rn1 =
{ rn2 = newrand rn1 ;
  v1 = if (rn2<0.0) then (-1.0) else 1.0 ;
  rn3 = newrand rn2 ;
  v2 = if (rn3<0.0) then (-1.0) else 1.0 ;
  rn4 = newrand rn3 ;
  rn5 = newrand rn4 ;
  (a,b) = if (rn5<0.0) then (0.0, v2) else (v2, 0.0);

  (x,y,z,vx,vy,vz) =
    if (rn4<(-0.33)) then (v1, 0.0, 0.0, 0.0, a, b)
    else if (rn4<0.33) then (0.0, v1, 0.0, a, 0.0, b)
                       else (0.0, 0.0, v1, a, b, 0.0);
 in
   (rn5,x,y,z,vx,vy,vz)
};

def randplanete am ar m0 r0 rn1 rap =
{ (rn2,xu,yu,zu,xvu,yvu,zvu) = directunit rn1 ;
  rn3 = newrand rn2 ;
  rn4 = newrand rn3 ;
  m = m0*(1.0 + am*rn3)/rap ;
  r = r0*(1.0 + ar*rn4) ;
  v = sqrt (m0/r) ;
in
```

```
  (rn4, (r*xu), (r*yu), (r*zu), (v*xvu), (v*yvu), (v*zvu), m, r)
};

def planete (x,y,z) (vx,vy,vz) am ar m0 r0 rnd nmax l stage stagemax rap =
{ rn1 = newrand rnd ;
  n = if (stage>stagemax) then 0
      else (truncate ((float nmax)*(rn1+1.0)/2.0)) ;
  (rn2, l2) = { for i <- 1 to n do
                  (rn, dx,dy,dz, dvx,dvy,dvz, m, r)
                    = randplanete am ar m0 r0 rn1 rap ;
                  (next rn1, next l)
                    = planete (x+dx, y+dy, z+dz) (vx+dvx, vy+dvy, vz+dvz)
                              am ar m (r/rap) rn nmax l
                              (stage+1) stagemax rap ;
               finally (rn1, l)
              };
in (rn2,
   ({record
       m = m0 ;
       x = x ;
       y = y ;
       z = z ;
       vx = vx ;
       vy = vy ;
       vz = vz ;
    }:l2) )
};

def initbodies (stagemax, nmax, am, ar, m0, r0, rap, rnd) =
{ (_,lst) = planete (0.0,0.0,0.0) (0.0,0.0,0.0) am ar m0 r0
              rnd nmax nil 1 stagemax rap ;
  nb = (length~list lst) - 1 ;

 in
   ((nb+1),{array (0,nb) of [i] = (nth~list i lst)
                         || i <- 0 to nb })
  };
```