Request for Comments : 493
Groupe de travail Réseau
NIC : 15358
Références : 282, 258
RFC rendue obsolète : 292
Traduction Claude Brière de L'Isle

J. Michener, MAC
I. Cotton, MITRE
K. Kelley, U. of Ill.
D. Liddle, Ownes Ill.
E. Meyer, MAC
avril 1973

Protocole GRAPHICS

Introduction

Le présent document reflète les opinions exprimées et les décisions adoptées à la seconde réunion du groupe Network Graphics, tenue au Laboratoire d'intelligence artificielle de Stanford University en novembre 1971. Il décrit une partie de la proposition de norme de protocole Graphics pour le réseau pour la transmission de données graphiques au sein du réseau ARPA. Les aspects particuliers du protocole couverts dans le présent document se rapportent à la forme et au contenu des informations graphiques envoyées d'une source d'informations graphiques (un programme d'application, par exemple, dans "l'hôte serveur") à un dispositif d'affichage en sortie sur une console graphique (chez "l'hôte utilisateur"). Cela va prendre la forme d'une séquence d'octets, et nous allons l'appeler le flux binaire graphique de sortie.

Le présent document est destiné à servir de base de discussion et d'expérience sur le réseau. Le présent document n'inclut pas la forme ou le contenu des entrées graphiques (les données envoyées de l'hôte utilisateur à l'hôte serveur) ni la façon dont la connexion est établie entre les hôtes. Une proposition sur la première question sera peut-être formulée par le présent comité ; la seconde relève du comité Connexion (du groupe Network Graphics).

La présente RFC décrit les commandes qui sont disponibles dans le protocole en termes d'effet qu'elles auraient à l'extrémité de réception (hôte utilisateur). Il est clair qu'un ensemble de logiciels est nécessaire du côté de l'hôte serveur pour que les applications puissent transmettre des données graphiques, mais ce sujet est en dehors du domaine d'application de la présente RFC.

Le lecteur pourra observer qu'aucune facilité n'est spécifiée dans le présent protocole pour permettre à l'hôte utilisateur de rapporter les erreurs logiques survenant dans le flux binaire de sortie graphique à l'hôte serveur. Une telle facilité devrait être intégrée dans le flux binaire d'entrée graphique, car elle implique la plupart des problèmes qui se rapportent à la synchronisation d'hôtes indépendants.

Fondements

Le lecteur devrait probablement lire attentivement la RFC 282: "Rapport de la réunion Graphics" de Mike Padlipsky pour replacer le cadre de la discussion sur les échanges de graphismes sur le réseau. Il pourrait aussi être intéressant de prendre note du modèle décrit dans la RFC 285 "Network Graphics" de Donald Huff.

Niveaux et règles de base qui en découlent

Au sein du protocole graphics les fonctions seront classées en un certain nombre de niveaux qui dépendent en partie de leur difficulté de mise en œuvre. Il est prévu que tout hôte qui revendique la mise en œuvre des fonctions de niveau N doit mettre en œuvre aussi tous les niveaux inférieurs. Et donc, il est envisagé que les sites mettent en œuvre les niveaux de façon incrémentaire. Les mises en œuvre seront améliorées selon un processus continu d'inclusion de plus en plus de fonctions, et il est prévu que chaque mise en œuvre soit capable d'identifier son propre niveau de protocole graphics auprès d'un site distant qui demande un échange graphique. Un résultat secondaire est que chaque site sera capable de déterminer ses propres priorités en engageant des programmeurs sur le protocole graphique par rapport à d'autres efforts.

Notre intention est aussi que la mise en œuvre du niveau N n'exige aucune connaissance du niveau N+1. Et donc un site peut mettre en œuvre un niveau en sachant (en principe) qu'aucun changement à des niveaux supérieurs ne va altérer le niveau mis en œuvre. Il peut être décidé à certains moments par le groupe Network Graphics de redéfinir un niveau qui avait été précédemment confirmé. Il n'est pas prévu que cela doive arriver mais il faut reconnaître que le protocole Graphics proposé est expérimental et peut devoir être changé.

Autre règle de base : un flux de commandes et de données qui est valide à un niveau donné K doit produire des résultats "identiques" sur tout interpréteur de niveau K ou supérieur. Cela veut dire que tant que les commandes et les données ne

tirent parti que des opérations strictement définies, il doit en résulter des images similaires. Les aspects du protocole qui ne sont pas strictement définis (pour l'instant) incluent la taille de caractère, la position relative du caractère par rapport à l'axe, la façon dont, dans la sortie de texte, les caractères de contrôle affectent le terminal, et ce qui se produit lorsque l'axe est déplacé ou qu'une ligne est tracée en dehors des limites logiques de l'écran. Cette règle force à la compatibilité aval, de sorte qu'une application écrite en utilisant les caractéristiques d'un niveau de faible numéro fonctionnera toujours sur des sites qui sont passés à la mise en œuvre de niveaux supérieurs. De plus, tous les aspects du présent protocole qui sont explicitement "laissés non spécifiés" dans les descriptions du fonctionnement détaillé ci-dessous devront être explicitement spécifiés dans toute description publique d'une mise en œuvre réelle.

Nous allons maintenant décrire le cadre commun à tous les niveaux.

Formes de base des données

Les informations dans le protocole de la norme Graphics du réseau seront exprimées comme des séquences d'octets. Une commande consistera en un octet de commande suivi par zéro, un ou plusieurs arguments. Le même octet de commande prendra toujours le même nombre d'arguments dans la même forme. La longueur de chaque argument peut être fixe ou variable selon l'argument.

Un type simple d'argument est une "valeur", qui est un entier de 8 bits. Un autre type d'argument est une "chaîne" qui est un compte suivi d'un nombre (le compte) d'octets de 8 bits. Si le compte est entre 0 et 127, il est envoyé dans un seul octet. Si le compte est entre 128 et 2**15-1 (**signifie l'exponentiation), il est envoyé sur deux octets avec le bit de poids fort du premier octet mis à un. Le premier octet contient les sept bits de plus fort poids et le second octet contient les huit bits de moindre poids. Une chaîne est le seul type d'argument d'une commande dont la longueur puisse varier. Par exemple, chaque fois qu'une commande a des arguments facultatifs, ils seront représentés à l'intérieur d'une chaîne.

Les données de coordonnées ont engendré de considérables discussions à la seconde réunion du groupe Network Graphics. Il a été décidé qu'un système de coordonnées logiques à deux dimensions était nécessaire, et chaque interpréteur du flux binaire de commande graphique serait chargé de transposer ce système de coordonnées en coordonnées de dispositif physique. Il a été décidé que les données dans le système de coordonnées logiques seraient en notation de complément à deux, qu'elles seraient fractionnaires, que chaque bord de l'écran aurait la longueur unitaire, et que l'origine correspondrait au centre de l'écran de l'appareil de sortie. Les bordures verticales (horizontales) de l'écran de l'appareil de sortie correspondent aux lignes X (Y) = -1/2 ou X=+1/2-e où e est un entier positif petit déterminé par la précision des données fractionnaires. En particulier les points (-1/2, -1/2) (-1/2, 1/2-e), (1/2-e, -1/2) et (1/2-e, 1/2-e) doivent être des points visibles aux coins de l'écran logique. (Dans le cas d'une surface d'affichage non rectangulaire, c'est le choix de la mise en œuvre qui les détermine.) Et donc, nous dirons que le système des coordonnées logiques contient les points dont les coordonnées sont dans la gamme de -1/2 à un petit peu moins que +1/2.

Les commandes qui prennent les données de coordonnées seront disponibles en divers modes. En mode absolu, une position est spécifiée en donnant ses coordonnées dans le système de coordonnées logiques. En mode relatif, la différence entre les coordonnées de la position et les coordonnées de la position actuelle doivent être spécifiées. Et donc une donnée de coordonnée qui est un argument pour une opération en mode absolu devrait être dans la gamme -1/2 à +1/2-e, alors que celle d'une opération en mode relatif devrait être dans la gamme -1+e à +1-e.

La seconde réunion du groupe Graphics a exprimé son intérêt pour une définition d'un très large espace de coordonnées (de nombreux bits de précision dans chaque coordonnée fractionnaire). Ceci sera fait en permettant que la longueur de chaque donnée de coordonnée soit établie comme un mode en octets de 8 bits. Il a été décidé à cette réunion que deux octets par coordonnée suffiraient pour l'instant. Et donc "e" dans la discussion ci-dessus est 2**(-15) (un dans le bit de moindre poids de la coordonnée fractionnaire d'un signe plus de 15 bits).

Les données de texte seront transmises comme un argument de diverses commandes à afficher sur l'appareil de sortie. L'ASCII du réseau sera utilisé pour représenter les caractères. Au plus bas niveaux du protocole, une seule taille de caractère sera disponible – sans considération de ce qui est "normal" sur l'appareil d'affichage. Si l'appareil n'a pas de taille "normale", 72 caractères par ligne serait souhaitable. Aux plus hauts niveaux, la taille de chaque caractère individuel peut être spécifiée.

Aussi, aux niveaux inférieurs, les caractères de contrôle seront passés à l'appareil d'affichage pour qu'il les traite au mieux de ses possibilités. Cependant, le consensus de la réunion Graphics s'est fait sur l'idée qu'à un certain niveau raisonnablement bas (mais pas zéro), retour chariot, saut à la ligne, et effacement arrière devraient être interprétés comme effectuant ces opérations.

Sous programmes d'images et sujets qui s'y rapportent

À la seconde réunion du groupe Network Graphics, il a été décidé que deux sortes de sous programmes d'images étaient souhaitables, la principale distinction entre elles étant la difficulté relative de mise en œuvre. À la réunion, la variété la plus simple était appelée une sous image, et la plus complexe un sous programme. L'auteur estime que ces termes ne véhiculent pas une sémantique suffisante pour qu'ils soient conservés et propose donc de normaliser à la place "simple sous programme" et "sous image complète".

Le seul paramètre qui puisse être passé à une simple sous image est la "position axiale réelle". En d'autres termes, si une telle sous image est appelée plus d'une fois dans une image, la seule différence d'apparence entre les diverses invocations est une traduction due à la position axiale au moment de l'invocation. D'un autre côté, les sous images complètes prennent des paramètres qui peuvent causer des mises à l'échelle, des rotations, des réflexion, ou tout autre transformation imaginable.

Il est prévu qu'une définition de sous-image n'a besoin d'être transmise qu'une seule fois (par connexion réseau) et ne serait pas supprimée par une opération de "nouvelle image". Et donc une image changeante pourrait être subdivisée en plusieurs parties sur la base des informations statiques et des informations changeantes ; seules les définitions des parties qui changent ont besoin d'être transmises pour redessiner l'image.

Traditionnellement, les sous-programmes d'images qui ne dépendent que de la position axiale initiale ont été restreints à des opérations de traçage en mode de données relatives. En considérant le fait que les sous-images seront probablement utilisées pour sauvegarder des informations d'image statique, il est souhaitable de permettre des opérations en mode de données absolues dans les sous-images simples.

La question suivante qui se pose naturellement est ce que signifient des données absolues dont une sous-image complète prend à la fois des paramètres de position et d'échelle ? Les données absolues sont elles réellement absolues dans ce cas ? L'auteur estime que la réponse est la suivante : la sous-image complète est réellement une image de plein droit, donc elle a son propre système de coordonnées logiques, et ses données absolues sont réellement au sein de ce système de coordonnées. Et donc "déplacer et changer d'échelle" une sous-image complète signifie réellement "dimensionner la sous-image dans son propre système de coordonnées et déplacer le résultat comme un tout".

En résumé, une différence majeure entre sous-image simple et complète est alors que la sous-image complète a son propre système de coordonnées logiques et qu'une sous-image simple utilise le système de coordonnées logiques de qui l'invoque. Cette distinction est la raison pour laquelle les sous-images complètes sont plus difficiles à mettre en œuvre que les sous-images simples.

Un autre point discuté à la réunion était un mode de données spécial dans lequel une sous-image peut afficher des données dans des positions absolues sur l'écran, c'est-à-dire, absolument dans le programme principal (d'image). Pour le réaliser, les participants à la réunion ont proposé des modes de données particuliers pour les trois opérations : déplacement axial invisible, traçage de ligne, et affichage de point. L'idée de ces modes de données est de court-circuiter toutes les fonctions de rotation, de changement d'échelle et de coupure, associées au niveau actuel d'incorporation de sous-image jusqu'à ce que ce mode soit nettoyé d'une certaine façon. Le même résultat peut être obtenu plus directement et mis en œuvre plus efficacement par deux commandes : une pour sauvegarder et une pou rétablir le système de coordonnées logiques pour la sous-image en cours. (De plus, l'opération de "sauvegarde" établirait bien sûr le système de coordonnées logique initial, de plus haut niveau.)

Invocations de sous-image simple

À côté de l'<identifiant> de la sous-image à invoquer, une simple invocation de sous-image peut spécifier deux paramètres facultatifs ; le premier est un <identifiant> qui est le "nom" (dans un sens qui est décrit plus loin) de cette invocation de sous-image particulière, et le second est une position absolue sur la page d'invocation à laquelle elle sera invisiblement déplacée, avant d'invoquer la sous-image. Lorsque (finalement) le spectateur est autorisé à interagir en "cueillant" les informations affichées devant lui, si les informations font partie d'une sous-image, le "nom" de l'invocation de sous-image fera alors partie de "l'entrée graphique" rapportée à l'hôte serveur. Si les informations cueillies par le spectateur sont dans plusieurs niveaux d'invocation de sous-image, les noms de chacune des invocations seront rapportés d'une façon qui indique la façon de les incorporer. (Noter que le seul nom de la sous-image n'est par lui-même pas suffisant, car une sous-image peut être affichée dans plusieurs positions et l'application peut souhaiter distinguer les invocations individuelles.) Si l'identifiant n'est pas spécifié, sa valeur par défaut est la chaîne nulle. Si la position (pour le déplacement invisible) n'est pas spécifiée, on utilise la position actuelle de l'axe.

Lorsque l'un ou l'autre de ces deux paramètres est présent, il est codé sur deux bits dans un octet de code qui précède les

paramètres. Si les deux paramètres sont présents, ils sont alors toujours dans le même ordre ; cet ordre et les bits de l'octet de code alloués aux deux paramètres sont spécifiés dans la description détaillée de la commande Simple Instance (et dans le BNF en Appendice 1). Un compte des données dans le reste de la commande d'instance précède l'octet de code, et suit immédiatement le nom de la sous-image qui est invoquée. Il est donc inclus de telle sorte qu'il n'est pas nécessaire de décoder l'octet de code pour déterminer la longueur totale de n'importe quelle opération Simple Instance.

Fenêtrage : Coupures, blanchiment, ou (?)

Pendant que nous en sommes aux systèmes de coordonnées et de sous-images, il peut sembler bon de toucher un mot du sujet de qui (quelle extrémité de la connexion) est chargé de faire quoi, lorsqu'une image est sur le point d'être affichée au delà des bordures de l'écran virtuel ? Le consensus de la réunion Graphics s'est fait sur l'idée que l'interpréteur du protocole graphique (c'est-à-dire, l'extrémité utilisatrice) ne devrait pas être chargé de faire quoi que ce soit de raisonnable dans le cas où une image affiche des informations au-delà de la bordure de l'écran (par exemple, par des déplacements et tracés relatifs).

L'interpréteur doit cependant réagir de façon appropriée aux prochaines données absolues dans la gamme appropriée. Dans les systèmes graphiques existants, les diverses solutions à cette situation incluent :
- de couper une ligne pour afficher autant qu'il est approprié,
- de blanchir la totalité d'une ligne si une partie en est invisible, ou
- d'éliminer les bits de plus fort poids du registre de position actuel, de façon qu'aucune position invisible ne puisse être représentée ("enveloppement").

En plus des problèmes d'effets de bordure au plus haut niveau, des problèmes surviennent par rapport aux sous-images (complètes). C'est bien agréable d'être capable de sélectionner une portion rectangulaire d'une sous-image à afficher au titre d'une invocation de sous-image. (Voir : Newman, Procédures d'affichage, Communications de l'ACM, volume 14, n° 10, octobre 1971, pages 651-660). Conformément au consensus de la réunion, qui était de rendre facultative cette capacité, l'auteur espère simplement inclure dans le protocole une méthode de codage de cette information dans la mesure où son site a) peut traiter un tel fenêtrage, et b) espère fournir une facilité de service pour effectuer cette fonction.

L'Appendice 2 décrit comment enchaîner plusieurs niveaux de portions dans un seul essai rectangulaire, pour autant qu'aucune rotation ne soit impliquée. Il souligne aussi les problèmes qui se rapportent aux rotations et portions.

Invocations de sous-image complète

Nous sommes maintenant en position de considérer ce qui peut être spécifié au titre d'une invocation de sous-image complète, en plus du nom de la sous-image invoquée, qui est, bien sûr, exigé. Les données décrites ci-dessous seront toutes facultatives : un seul octet de code va précéder toutes ces données ; la présence ou l'absence de l'un des paramètres sera indiquée par un bit dans l'octet de code qui sera à un ou zéro. Les paramètres vont toujours apparaître dans le même ordre, si ils sont présents. Cet ordre est donné ci-dessous dans la description détaillée de la commande Full Instance (et dans le BNF de l'Appendice 1). De plus, précédant même l'octet de code, il y aura un <compte> des octets qui suivent, y compris l'octet de code pour déterminer la longueur totale de toute opération particulière de Full Instance.

Un paramètre est un <identifiant> qui peut être utilisé pour distinguer cette invocation particulière de cette sous-image de toutes les autres invocations de la sous-image. Ce paramètre a déjà été décrit dans les invocations de simple sous-image.

Un paramètre qui peut être spécifié est une translation : cela sera spécifié en donnant les coordonnées absolues du centre (sur la page d'invocation) de l'image de la sous-image ; cela sera par défaut la position axiale actuelle au moment de l'invocation.

Une rotation peut être spécifiée en donnant une fraction de 16 bits dans la gamme de 0 à .1111111111111111 (binaire) inclus ; cette fraction représentera la partie d'un cercle complet (2 pi) de la rotation. La valeur par défaut de l'angle de rotation sera zéro.

(En réalité, le schéma de représentation de la rotation fonctionne de la même façon que si on la voit comme fraction d'un complément à deux de -1/2 à juste un peu moins que +1/2. C'est-à-dire que la même configuration binaire code la même rotation, du fait de la nature périodique du sinus et du cosinus. Par exemple, le zéro binaire représente toujours 0 pi ; 010000...0 note pi/2 dans les deux schémas ; 100...00 note 1/2 dans un schéma et -1/2 dans l'autre, ce qui correspond à des rotations respectivement de +pi et -pi, c'est-à-dire des rotations identiques.)

Une portion rectangulaire de l'image invoquée est aussi spécifiable au titre d'une sous-image complète à figurer sur l'image

invoquée (voir au paragraphe précédent l'exposé sur les coupures). Ce rectangle est spécifié par son centre et une moitié de sa taille totale en x et y. C'est à dire que le rectangle va comporter tous les points dont les coordonnées en x diffèrent de celles du centre de pas moins que la taille spécifiée en x et celles dont les coordonnées en y satisfont une condition similaire. La position par défaut pour ces valeurs placera le centre à l'origine et donnera aux deux demi largeurs x et y la valeur de +1/2. Et donc la valeur par défaut comporte la totalité du système de coordonnées logiques de la page invoquée (et aussi certains points dont les coordonnées sont +1/2, ce qui, strictement parlant, est "en-dehors" du système de coordonnées ; comment cette incohérence est résolue est non spécifié).

Finalement, on doit spécifier l'échelle à appliquer pour déterminer l'image ; cela peut être fait de nombreuses façons. L'une d'elles est de spécifier un grossissement uniforme à appliquer à la sous-image. Pour qu'une large gamme de grossissements puisse être réalisée, l'auteur estime qu'il faudra utiliser une forme de notation scientifique (c'est-à-dire à virgule flottante). Si il y a déjà une notation de virgule flottante normalisée dans le réseau (ce qui n'est pas le cas à ma connaissance) elle devrait être employée. Faute de quoi, il est suggéré que cette notation comporte un exposant de 8 bits (en complément à deux) suivi par une partie fractionnaire de 16 bits (en complément à deux).

Une autre forme d'échelonnage est de spécifier un grossissement séparé en x et en y, à appliquer à la sous-image avant d'effectuer toute rotation. Une troisième façon est encore de spécifier une zone rectangulaire dans le système de coordonnées de l'image invoquée qui sera rempli avec l'image de la sous-image. Comme le centre de l'image est déjà spécifié (par la translation), ces informations d'image comportent seulement les données de taille de la demi bordure. Si aucune des trois méthodes d'échelonnement n'est choisie (et qu'une transformation affine (voir ci-dessous) n'est pas donnée explicitement), on utilisera alors un grossissement uniforme de l'unité(c'est-à-dire, pas de mise à l'échelle). Noter que les trois formes de mise à l'échelle tendent à se contredire l'une l'autre et une seule d'entre elles devrait être utilisée dans une invocation donnée. Ce qui arrive si des informations contradictoires sont données dans ces champs est non spécifié.

L'Appendice 2 présente les mathématiques impliquées dans la transformation du système de coordonnées de la sous-image dans le système de coordonnées de l'invocation de l'image. Il montre que toutes les opérations individuelles (mise à l'échelle, rotation, et translation) peuvent être représentées comme une seule transformation affine (qui consiste en 6 valeurs). Il peut être agréable de permettre au programme de service de spécifier cette transformation directement. En conséquence, un paramètre possible d'une invocation de sous-image complète consistera en six nombres en virgule flottante (de la forme décrite au paragraphe grossissement, ci-dessus) à interpréter comme une transformation affine. Bien sûr, si la transformation affine est de la forme suivante :

```
/_ |x  |y_/ = /_ x y_/ * / L11 L12  / + /_ T1 T2_/   /_ L21 L22 _/
```

les valeurs devront alors être envoyées (arbitrairement) dans l'ordre (par colonnes) suivant : L11, L21, L12, L22, T1, T2. Cette transformation affine devrait être réversible, c'est à dire que L11*L22 - L21*L12 devrait être différent de zéro.

Affichage d'image

Un autre sujet abordé à la réunion et soumis pour décision au comité du protocole était la capacité de placer l'image de "niveau supérieur" dans un rectangle de l'écran virtuel. Le rectangle par défaut pourrait être l'écran plein. Autrement, il pourrait être laissé à la décision du visionneur de spécifier la position par défaut, via une interaction avec le système graphique chez l'hôte utilisateur. En général, l'affichage d'image permet de voir plus d'une image de "niveau supérieur" à la fois. Le désir de voir plusieurs images différentes sur le même écran arrive dans les cas où plusieurs utilisateurs travaillent ensemble et dans les cas où un utilisateur interagit avec un groupe d'applications (chez des hôtes serveurs distincts). L'auteur soutient que les transformations coordonnées exigées par ce dispositif sont plus simples que celles de la "sous image pleine" car il n'implique aucune rotation, et ferait partie du même mécanisme dans sa mise en œuvre. En particulier, simplement une autre transformation affine (voir l'Appendice 2) serait ajoutée aux niveaux causés par les invocations de sous-image pleine. Tout cela exige qu'on garde trace des identifiants d'accès d'image et des rectangles associés. Comme cela n'implique que peu de travail supplémentaire, il est proposé que ce dispositif soit inclus à un haut niveau du protocole.

Codes de commande

Chaque commande du protocole graphique se verra allouer une valeur non négative qui représentera cette commande dans le flux d'octets. L'algorithme par lequel les valeurs sont associées aux commandes est une question très délicate. Il y a cinq ou dix critères différents pour un "meilleur" algorithme, chaque critère insistant sur un angle d'approche différent. Ce nœud gordien sera tranché, dans cette proposition, en ordonnant les commandes approximativement selon les niveaux, puis en les numérotant. De plus, si plusieurs commandes en relation étroite surviennent au même niveau, il sera essayé de coder les variations de signification en termes de configurations binaires. Même si des considérations ultérieures amènent à proposer un changement de l'ordre, le sentiment du comité est que la numérotation ne devrait pas être altérée. Cependant, tant que le

sujet n'est pas définitivement tranché, il est vivement recommandé que toute mise en œuvre tienne compte de la possibilité de réallocation des codes de commandes.

Proposition particulière pour le protocole de niveau 0

Il est proposé que le niveau 0 reste très simple. Cela de façon à ce que la mise en œuvre se fasse rapidement et que l'expérimentation commence avec le protocole. Une autre raison est que l'hôte le moins puissant et même les terminaux programmables devraient être capables de le mettre en œuvre. En conséquence de cela, la "règle" a été établie que ne soit incluse une commande que si son résultat est une fonction uniquement de la commande en cours et la "position axiale" actuelle au début de la commande. En d'autres termes, l'interprète pour le niveau 0 n'a besoin d'aucune capacité interne de mémorisation pour les "modes" ou piles de commandes. Grâce à cette restriction, on espère qu'une mise en œuvre très simple sera possible pour le niveau 0. En particulier, peut-être qu'on pourrait même construire un traducteur matériel à partir du code de niveau 0 sur un code de terminal particulier.

Noter que dans l'allocation de opcode pour le niveau 0, les bits 4, 2, et 1 ont une signification spéciale pour les commandes mouvement, ligne, et point. En particulier, le bit 1 code le mode de données absolues contre le mode de données relatives, le bit 4 code si un résultat visible survient, et le bit 2 détermine si le résultat visible est une ligne ou un point.

Niveau 0 : résumé des commandes

Ci-après figure une liste de commandes (et leur syntaxe) au niveau zéro. Une description détaillée de ces commandes figure au paragraphe suivant. Les commandes qui traitent du protocole pourront être ajoutées par le comité de connexion. (Elles couvrent actuellement les opcodes dans la gamme 128 à 255.)

(Comme décrit dans Formes des données de base, ci-dessus, <x coordinate>, <y coordinate>, <x delta> et <y delta> sont des valeurs de coordonnées de deux octets, <string> est un compte suivi par <count> du nombre d'octets et <value> est un nombre de huit bits.)

Décimal	Octal	Binaire	Format
0	0	00000000	Nul
1	1	00000001	Écrase l'écran et replace l'axe
2	2	00000010	Mouvement absolu de <x coordinate> <y coordinate>
3	3	00000011	Mouvement relatif de <x delta> <y delta>
4	4	00000100	Trait absolu de <x coordinate> <y coordinate>
5	5	00000101	Trait relatif de <x delta> <y delta>
6	6	00000110	Point absolu de <x coordinate> <y coordinate>
7	7	00000111	Point relatif de <x delta> <y delta>
8	10	00001000	Chaîne de texte <chaîne>
9	11	00001001	Chaîne de TextR <chaîne>
10	12	00001010	Fin d'image
11	13	00001011	Échappement de <valeur> <chaîne>

Niveau 0 : Description des commandes

0 Déclaration Nul ("NULL").
Cette déclaration n'a pas d'argument – et pas d'effet non plus.

1 Écraser l'écran et replacer l'axe à l'origine ("ERASE").
Cette commande indique qu'une nouvelle image va être dessinée. Elle devrait toujours être couplée (en fin de compte) avec une commande Fin d'image ultérieure.

2 Déplacement invisible de l'axe à une position absolue
("MOVEA") <x coordinate> <y coordinate>.
Rien n'est tracé ; l'axe est positionné à la position absolue spécifiée de coordonnées x,y.

3 Déplacement invisible de l'axe d'une quantité relative
("MOVER") <x coordinate> <y coordinate>.
Rien n'est tracé ; l'axe est translaté de la quantité spécifiée en x et y.

4 Tracer une ligne à une position absolue
("DRAWA") <x coordinate> <y coordinate>.
Une ligne est tracée à partir de la position actuelle de l'axe jusqu'à la position absolue spécifiée de coordonnées x,y.

5 Tracer une ligne à une position relative
("DRAWR") <x delta> <y delta>.
Une ligne est tracée à partir de la position actuelle de l'axe jusqu'à la position delta x et delta y.

6 Afficher un point à la position absolue
("DOTA") <x coordinate> <y coordinate>.
L'axe est déplacé de façon invisible à la position absolue x,y et un point est affiché là.

7 Afficher un point à la position relative
("DOTR") <x delta> <y delta>.
L'axe est déplacé de façon invisible de la quantité spécifiée dans x et y et un point est affiché là.

8 Afficher une chaîne de texte ("TEXT") <chaîne>.
À la position actuelle de l'axe, afficher des caractères à la taille normale pour l'appareil concerné. <chaîne> consiste en un <compte> suivi par le compte du nombre de caractères. Si il n'y a pas de "taille normale", choisir la taille qui permet d'afficher soixante douze (septante deux) caractères par ligne. Les caractères dans la chaîne sont codés en US-ASCII. Tous les codes entre 0 et 127 (décimal) inclus sont permis. (Au niveau zéro, ce qui arrive aux caractères de contrôle n'est pas spécifié.) Lorsque, à la suite de l'exécution de cette commande, l'axe reste non spécifié, la prochaine commande d'affichage de texte qui suit immédiatement va ajouter son texte à la chaîne précédente. (L'utilisation de la commande TEXT est déconseillé ; utiliser plutôt la commande TextR.) La position du premier caractère par rapport à la position initiale de l'axe est laissée non spécifiée.

9 Affiche le texte et restaure l'axe ("TEXTR") <chaîne>.
À la position actuelle de l'axe, affiche une chaîne de caractères à la taille normale pour l'appareil en fonctionnement, puis repositionne l'axe où il se trouvait avant la commande. La <chaîne> consiste en un <compte> suivi par le compte des caractères. Si il n'y a pas de "taille normale", choisir la taille de façon que soixante douze caractères soient affichés par ligne. Les caractères dans la chaîne sont codés en US-ASCII ; tous les codes entre 0 et 127 (décimal) inclus sont permis. (Au niveau zéro, ce qui arrive aux caractères de contrôle est laissé non spécifié.) La position du premier caractère par rapport à la position initiale de l'axe est laissée non spécifiée.

10 Fin d'image ("ENDPIC").
Cette commande note la fin d'une nouvelle image. Elle doit être couplée à une commande ERASE précédente.

11 Échappement vers <valeur> <chaîne> ("ESCDEV") spécifique de l'appareil.
Si "valeur" est le code alloué (par le comité du protocole) à l'appareil en fonctionnement, transmettre alors les octets de <chaîne> (qui commencent par un <compte> indiquant le nombre d'octets) à l'appareil sans les examiner. Autrement, ignorer cette commande. Si l'appareil n'accepte pas les informations sur huit bits, reformater les données d'une façon conforme aux spécifications de l'appareil ; un exemple serait de jeter le bit de poids fort pour un appareil à sept bits, ou peut-être de rassembler cinq octets en un mot de 36 bits, en éliminant encore une fois les bits de poids fort. L'action des octets dans la chaîne devrait laisser (ou au moins restaurer) tous les registres matériels de position de l'axe dans l'appareil dont pourrait vraisemblablement dépendre l'interpréteur.

En réalité, cette commande ne devrait pas être utilisée. Elle a été incluse au niveau 0 afin que des applications spécifiques puissent faire des réglages de mode et autres manipulations spécifiques de l'appareil. Par exemple, les terminaux ARDS peuvent avoir en option plusieurs portées de sorties adressables de façon indépendante. Le mécanisme de sélection ne change l'état que quand une séquence particulière de caractères ASCII atteint le terminal. Et donc ESCDEV serait utilisé pour choisir quelle ou quelles portées doivent être affectées par les commandes suivantes. (L'état actuel est invisible au paquetage graphique chez l'hôte utilisateur.)

De plus, supposons qu'un autre fabricant de terminaux ait une option similaire, qui réponde à une séquence de code différente. Cette possibilité est le motif de cette ignorance conditionnelle de la commande ESCDEV sur la base de la "<valeur>" spécifiée. Étant donné qu'une application particulière ne va être utilisée que pour faire la sortie soit sur le ARDS soit sur le second matériel (avec l'option de portées multiples) l'application pourra alors toujours envoyer deux commandes ESCDEV, une applicable aux seuls terminaux ARDS et l'autre aux seuls seconds matériels.

Niveau 1

*Régler le mode de ligne ("LINMOD") <valeur>.
Cette commande règle le mode de ligne en cours parmi les modes possibles et la <valeur> qui établit chacun d'eux est : solide (0), tirés (1), pointillés (2), et les autres (3 ou >). Au début d'une nouvelle image (c'est à dire, après une commande Erase et une commande Reset), le mode de ligne est solide. Si un site n'a pas un certain mode directement disponible, il peut a) le simuler dans un logiciel, b) lui en substituer un autre (tirés à la place des pointillés, ou vice versa) c) l'ignorer entièrement. Ce qui est fourni devrait être clairement indiqué dans tout document public. Il est vivement recommandé qu'au moins l'état solide et un autre mode soient fournis.

*Régler l'intensité ("SETINT") <valeur>.
Cette commande règle l'intensité des lignes, points et caractères affichés à la suite de la commande. Si <valeur> est 128 en décimal, l'intensité normale devrait être établie. Si <valeur> est 255 en décimal, le plus brillant devrait être sélectionné, et si c'est 0, l'axe devrait être blanchi. Les valeurs intermédiaires devraient être transposées de façon appropriée à la discrétion de la mise en œuvre. Par exemple, si le plus brillant est le même que le normal, toutes les valeurs de 128 à 255 devraient être transposées en normal. Les informations affichées entre le début d'une nouvelle image (commande ERASE) et la première commande SETINT apparaissent à l'intensité normale.

*Sortie de texte ("TEXTO") <chaîne>.
À partir de la position en cours de l'axe, cette commande affiche la <chaîne> (de caractères US-ASCII) formatée comme si c'était de la frappe (à l'intensité en cours). <chaîne> consiste en un <compte> suivi par le nombre de caractères du compte. C'est à dire que le texte qui s'étend après la marge de droite sera cassé et repositionné à la marge gauche sur la ligne suivante. Des caractères de contrôle, seuls retour chariot, saut à la ligne, et espace arrière doivent être interprétés correctement.

*En-tête de sous image ("SUBHED") <identifiant> <compte> <info d'en-tête>.
Cette commande commence la définition d'une sous-image nommée "<identifiant>". Cette définition est terminée par une commande correspondante SUBEND. Cette définition sera conservée jusqu'à ce qu'une nouvelle soit spécifiée ou jusqu'à ce que la connexion réseau de graphics soit interrompue. Noter que <identifiant> est une <chaîne> qui consiste seulement en lettres majuscules et en nombres.

Les définitions de sous-image peuvent être incorporées, ce qui sera équivalent à transmettre séparément les deux définitions. En d'autres termes, tous les noms de sous-image sont globaux et sont "connus" de toutes les autres sous-images. Si une définition de sous-image n'a pas été reçue avant son utilisation dans une image, la sous-image vide devrait être affichée à sa place jusqu'à réception d'une définition.

Une définition de sous-image n'a pas besoin d'être transmise au titre d'une image (c'est-à-dire au sein d'une paire de commandes ERASE et END). Bien sûr, toutes les définitions de sous-image peuvent précéder l'image principale.

Actuellement, le <compte> sera toujours 1, ce qui indique que seulement un octet de <info d'en-tête> suit, mais à des niveaux supérieurs du protocole il peut être nécessaire d'avoir de la place pour une expansion. Dans les <info d'en-tête>, le bit 80 hexadécimal sera mis si cette sous-image peut être une simple sous-image, et le bit 40 hex sera mis si la sous-image peut être une sous-image complète. (Il est possible qu'une sous-image soit les deux.)

Les autres informations qui peuvent éventuellement être présentes dans <info d'en-tête> incluent de savoir si la valeur actuelle d'un certain mode ou paramètre devrait être sauvegardée en entrée, et restaurée en sortie, de ce sous-programme chaque fois qu'elle est invoquée. Ces modes et paramètres incluent : mode de ligne, intensité, taille de caractères, et longueur des données.

*Fin de sous-image ("SUBEND").
Cette commande termine la définition d'une sous-image. Chaque SUBEND doit correspondre à la commande SUBHED précédente.

*Simple instance ("INSTS") <identifiant> <queue de simple instance>
Cette commande indique que l'<identifiant> de sous-image est à invoquer (instancier). À ce niveau, le niveau 1, aucune sous-image ne peut en invoquer une autre ; si on le fait, ce qui arrive reste non spécifié. Aussi, cela doit être une invocation d'une sous-image simple. Et donc, le bit 80 hex du seul octet de <info d'en-tête> doit avoir été mis dans la commande SUBHED qui débute la définition de <identifiant>. Si <identifiant> de sous-image n'a pas été défini, la sous-image vide devrait être affichée à la place.

La <queue de simple instance> commence par un compte de la quantité d'informations qui suivent. Ce compte peut être zéro. Si il est différent de zéro, le prochain octet est un octet de code à interpréter pour voir quelles autres informations

suivent. Si le bit 80-hex est mis, ce qui suit dans le flux d'octets est un <identifiant> (appelé "AS information"). Cet <identifiant> est le nom de cette instance particulière de la sous-image, comme décrite sous Invocation de simple sous-image. Si le bit 40-hex est mis, ce qui suit dans le flux d'octet (à la suite de AS information, si elle est présente) est une position x,y (dans le schéma de coordonnées de l'image appelante) à laquelle la sous-image sera centrée. (C'est appelé AT information.)

Si AT information n'est pas spécifié, la position actuelle de l'axe est utilisée par défaut. Si AS information n'est pas spécifié, elle est par défaut à la <chaîne> contenant zéro caractère. Si ni le bit 40 hex ni le bit 80 hex ne sont mis, ni le AT information ni le AS information ne sont présents, et l'octet de code devrait être zéro. (Aussi, le compte de longueur ferait mieux d'être 1.)

Changement des commandes de niveau 0 pour le niveau 1.

TEXT et TEXTR – Les caractères Retour chariot, saut de ligne et espace arrière devraient absolument être interprétés chaque fois qu'ils apparaissent dans une <chaîne>. Le résultat des autres caractères de contrôle reste non spécifié. L'intensité des caractères sera affectée par la commande SETINT.

ERASE – L'intensité normale et le mode ligne solide doivent être établis au début d'une nouvelle image.

DRAWA et DRAWR – Le mode de ligne et l'intensité doivent être affectés par les commandes LINMOD et SETINT.

DOTA et DOTR – L'intensité doit être affectée par la commande SETINT.

Niveau 2

*Marque ("MARK").
Cette commande cause la sauvegarde de la position courante de l'axe x,y sur une pile descendante. Cette pile descendante doit être distinguée de la pile descendante d'invocation de sous-image.

*Aller à la marque et monter ("MOVEMK").
Cette commande règle la position actuelle de l'axe égale à la position x,y au sommet de la pile de "marque" descendante. Si la pile est vide, on utilise l'origine à la place. Puis on fait apparaître la pile (sauf si elle est vide).

*Tracer jusqu'au sommet de la marque ("DRAWMK").
Si la pile de "marque" descendante n'est pas vide, cette commande trace une ligne (de mode et d'intensité de ligne courants) à partir de la position actuelle de l'axe jusqu'à la position x,y au sommet de la pile de "marque" descendante, et règle la position de l'axe à cette valeur. Puis on fait apparaître la pile. Si la pile est vide, la ligne est tracée jusqu'à l'origine et la position de l'axe est aussi établie à cet endroit.

Changement du niveau 0 et 1 au niveau 2.

INTS – les niveaux arbitraires de sous-images simples doivent être pris en charge.
(Noter que l'utilisation récurrente des sous-images n'est pas permise : une fois que la récurrence commence, elle ne peut plus être arrêtée.) La pile descendante pour les invocations de sous-images doit être gardée à part de la pile de "marque" descendante.

Niveau 3

(Peut-être que les transformations de rotations devraient être mises à un niveau supérieur, par exemple plus haut que les opérations d'accès à la vue.)

*Instance pleine ("INSTF") <identifiant> <queue d'instance pleine>
Cette commande indique que <identifiant> de sous-image doit être invoqué (instancié) de manière "pleine" comme décrit dans une section explicative. Pour une chose, cela signifie que le bit 40 hex du seul octet de <info-d'en-tête> doit avoir été mis dans la commande SUBHED qui débute la définition de <identifiant>. Si <identifiant> n'a jamais été défini, la sous-image vide (c'est-à-dire, rien) devrait être affichée à sa place.

La <queue d'instance pleine> est similaire à la <queue de simple instance> décrite sous la commande INSTS, mais elle contient plus d'informations. Ci-dessous figure une liste des informations qui peuvent être spécifiées, et du bit alloué à la présence/absence de chaque élément d'information. Les éléments d'information qui sont présents apparaissent toujours dans

le flux d'octets dans l'ordre dans lequel ils sont décrits dans cette liste. (Tous les éléments d'information sont décrits plus en détails dans Invocations de sous-image complète, sauf pour les "AS information" qui sont décrites dans Invocations des sous-image simple.)

Bit (hex)	Information
80	Comme information --"nom" de cette instance particulière. Consiste en un <identifiant>.
40	Information de traduction -- Centre de l'image de la sous-image sur la page d'invocation. Consiste en un <x coordinate> et un <y coordinate>.
20	Rotation – Partie fractionnaire de 2π pour faire tourner l'image dans le sens trigonométrique. Consiste en une fraction de 16 bits non signée.
10	Informations de portion – Partie rectangulaire d'une sous-image à afficher. Consiste en <x coordinate>, <y coordinate>, <x delta>, et <y delta>.
8	Grossissement uniforme – Proportion applicable à l'image complète. Consiste en un nombre à virgule flottante (qui ne devrait pas être zéro).
4	Grossissement séparé sur x et y – Des échelles différentes pour les axes x et y de la sous-image. Consiste en deux nombres à virgule flottante (dont aucun ne devrait être zéro).
2	Taille d'image – Dimensions du rectangle qu'occupe l'image sur la page d'invocation. Consiste en un <x delta> et un <y delta> (dont aucun ne devrait être zéro).
1	Transformation affine – La transposition du système de coordonnées appelé en celui appelant. Consiste en six nombres à virgule flottante.

Notes :
1) Au plus un des trois bits : 8, 4, et 2, devrait être mis.
2) Si le bit 1 est mis, les bits 2, 4, 8, 20 et 40 ne devraient pas être mis.
3) Si jamais des paramètres facultatifs supplémentaires sont ajoutés à l'invocation de sous-image complète, un autre octet de code pourrait suivre toutes les informations ci-dessus. Dans ce cas, la partie <compte> de la <queue d'instance complète> inclurait ce second octet de code et tous octets d'information supplémentaires.

*Échappement sur le système de coordonnées de niveau supérieur ("ESCTOP").
Jusqu'à ce qu'une commande RESLEV soit (ultérieurement) exécutée, toutes les commandes d'affichage (déplacements, traits, points, et textes) doivent fonctionner comme si elles étaient produites par l'image de niveau supérieur (principale) au lieu de la sous-image qui les contient. C'est-à-dire qu'elles doivent être transposées sur l'écran conformément à la transposition pour le niveau supérieur. Les invocations de sous-image elles-mêmes, qui sont faites alors qu'une commande ESCTOP est en cours, ne sont pas affectées par la commande. C'est-à-dire que les transformations sont calculées comme si la commande n'était pas effectuée. Les transformations calculées sont cependant ignorées, et les informations affichées par la sous-image apparaissent toujours comme étant au niveau supérieur, jusqu'à ce qu'une commande RESLEV annule le mode ESCTOP. Et donc, une invocation de sous-image exécutée alors qu'une commande ESCTOP est effectuée, agit comme si une commande RESLEV était exécutée immédiatement avant l'invocation, et qu'une commande ESCTOP était exécutée comme première commande de la sous-image. Des considérations similaires tiennent pour revenir des sous-images.

*Reprise du système de coordonnées du niveau actuel ("RESLEV").
Cette commande restaure le système de coordonnées logiques correspondant à la sous-image en cours d'exécution, dans le cas où ce système de coordonnées a été désactivé par une commande ESCTOP. (Voir à ESCTOP.)

Change les niveaux 0, 1 et 2 en niveau 3.

MARK – la position sauvegardée de l'axe doit être exprimée selon le système de coordonnées logiques, et non dans le système de coordonnées physiques.

TEXTR, TEXT, TEXTO – Comme une sous-image complète est supposée être transformée comme un tout, comme si elle était une image de plein droit, il paraît à l'auteur que, en particulier, tous les mouvements d'axe qui se rapportent aux caractères devraient être affectés. Cela inclut la taille des caractères, les tabulations, retour chariot et saut en fin de ligne. En particulier, le retour chariot devrait placer l'axe à la marge gauche ; c'est à dire au bord gauche du système de coordonnées logiques de la sous-image invoquée. Tous ces changements peuvent être très difficiles à accomplir, et ce qu'il convient de faire restera non spécifié pour le moment, et les lecteurs sont particulièrement invités à commenter.

Niveau 4

(Peut être que le point de vue du fonctionnement pourrait être inclus au niveau 3.)

*Declare Viewport

("SETVW") <viewport id> <x coordinate> <y coordinate> <x delta> <y delta>
Règle l'angle de vue identifié par <viewport id> pour représenter la zone indiquée de l'écran logique. Les données x et y ne sont pas les coordonnées de l'écran physique, car cela impliquerait une dépendance à l'appareil. Cette commande supplante complètement toute déclaration précédente du même angle de vue. Si les informations sont déjà affichées au sein de l'angle de vue spécifié, cette commande cause la relocalisation des informations affichées sur l'écran à leur nouvelle position.

Si la zone spécifiée excède les limites de l'écran standard d'affichage de graphique, ce qui arrive n'est pas spécifié. Les angles de vue ne sont pas nécessairement disjoints ; en d'autres termes, deux angles de vue peuvent présenter des informations d'affichage au même point sur l'écran.

Si <x delta> ou <y delta> sont négatifs, l'angle de vue désigné devrait être supprimé. Toutes les informations qu'il affichait ne doivent plus apparaître.

Parce que cela affecte l'image de niveau supérieur, l'auteur pense que cette commande ne devrait pas intervenir au titre d'une image ou d'une déclaration de sous-image.

*Ajouter la sous-image à l'angle de vue ("ADDSVW") <identifiant> <identifiant d'angle de vue>
La sous-image nommée <identifiant> est affichée au sein de l'angle de vue spécifié, si il n'y est pas déjà affiché. (Si il l'est, rien n'est fait.) La sous-image doit être capable d'être invoquée via une invocation de sous-image complète. Si l'angle de vue n'a jamais été déclaré via une commande SETVW, ce qui arrive n'est pas spécifié. (Les trois possibilités sont : rien n'est affiché ; l'angle de vue par défaut est tout l'écran logique ; le spectateur humain peut avec Using Host spécifier l'angle de vue.) Si l'angle de vue est ultérieurement déclaré, la sous-image doit être affichée dedans. Si la sous-image n'a jamais été déclarée, rien n'est affichée pour elle ; lorsque et si elle est déclarée ultérieurement, la nouvelle définition est affichée dans l'angle de vue. Plus d'une sous-image peut être affichée en une seule fois dans un seul angle de vue.

Parce que cela affecte l'image de niveau supérieur, l'auteur estime que cette commande ne devrait pas intervenir au titre d'une image ou dans une déclaration de sous-image.

*Retirer l'angle de vue ("CLVW") <identifiant d'angle de vue>
Toutes les sous-images qui ont été ajoutées avec la commande ADDSVW à l'angle de vue spécifié dans cette commande en sont retirées. Et donc l'angle de vue spécifié ne contribue en rien à ce que voit le spectateur humain. (Après une commande CLVW, la zone de l'angle de vue peut n'être pas blanche du fait d'autres angles de vue, non retirés qui la chevauchent.)

Parce que cela affecte l'image de niveau supérieur, l'auteur estime que cette commande ne devrait pas intervenir au titre d'une image ou dans une déclaration de sous-image.

Change les niveaux 0, 1, 2 et 3 en niveau 4.

ERASE – Tous les angles de vue sont éliminés (comme dans la commande CLVW) mais leurs déclarations sont conservées.

ENDPIC -- Cette commande perd partiellement son objet : elle ne sert plus à marquer la fin de toutes les informations d'image à présenter à l'utilisateur, car des opérations d'angle de vue peuvent suivre qui amendent ou altèrent l'image. Cette fonction est partiellement remplie par les commandes DELAY et NODELAY décrites ci-dessous.

Niveau ?

*Établir la taille de caractère ("SETCHS") <x delta> <y delta>.
Sauf mention contraire, les caractères doivent être affichés de telle sorte que chacun occupe approximativement <x delta> et <y delta> dans la direction de coordonnées appropriée dans le système de coordonnées logiques actuel. L'espacement inter caractère et d'interligne pourrait être un certain pourcentage (des idées ?) en plus de <x delta> et <y delta>, ou il pourrait être spécifié séparément. Dans tous les cas, seul un "effort au mieux" pourrait être espéré sur un site. La taille de caractère est toujours réglée à normal (comme défini au niveau 0 avec la taille de caractère normale) par la commande ERASE. <x delta> et <y delta> devraient être positifs, sauf que si <x delta> est égal à zéro, <y delta> étant négatif, zéro, ou positif correspond à une taille de caractère qui est respectivement "inférieure à la normale", "normale", ou "supérieure à la normale". De combien supérieure ou inférieure à la normale est laissé à l'appréciation du site.

Changement des niveaux 0 et 1 en niveau ?.

TEXTR, TEXT, et TEXTO – les caractères sont à afficher conformément) la taille de caractère actuelle.

ERASE -- Doit établir la taille de caractère normale, normal étant celle du niveau 0.

Niveau ?'

*Régler la longueur des données ("SETDLN") <valeur>.
Jusqu'à ce que ce mode soit explicitement changé par une autre commande SETDLN, les diverses données vont comporter le nombre d'octets <valeur>. <valeur> peut être 1, 2, 3 ou 4. Les types syntaxiques suivants sont affectés (se référer à l'Appendice 1) : <coordonnée>, <coordonnée x>, <coordonnée y>, <double coordonnée>, <x delta>, <y delta>, <angle>, et la partie fraction d'un nombre à virgule flottante. Lorsque une connexion réseau est initialement établie, la longueur des données est deux.

Niveau ?"

(Ces commandes devraient probablement être au même niveau que les opérations d'angle de vue, sinon plus tôt.)

*Des changements étendus suivent ("DELAY").
Cette commande facultative est conçue pour éliminer des efforts inutiles de la part des programmes Using Host. Sur certains hôtes et/ou avec certains appareils de sortie (en particulier des tubes de mémorisation) une quantité non négligeable de temps peut être nécessaire pour présenter une image au spectateur humain. Si des changements étendus vont être faits, cette commande sera utilisée pour empêcher le paquetage graphique Using Host de mettre à jour l'image après chaque changement. Une commande NODELAY sort du mode DELAY et cause la préparation et la présentation de l'image au spectateur.

Par exemple, l'image actuelle peut afficher quatre sous-images dont chacune va être redéfinie. Sans une commande DELAY, le spectateur verrait les stades successifs du changement, chacun pouvant impliquer une grande quantité de calcul ou de temps de transmission.

*Fin de changements étendus ("NODELAY")
Cette commande facultative défait les effets de la commande DELAY.

Appendice 1 BNF pour le flux binaire du protocole Graphics

Clés de lecture :
Les non terminaux sont représentés entre < >.
Les terminaux qui sont des mots clés signifiant des valeurs particulières d huit bits sont en majuscules.
Les terminaux dont la signification devrait être claire pour le lecteur sont en minuscules.
Noter que "empty_string" signifie "zéro octet", et non "une <chaîne> dont le <compte> est zéro."

```
<flux d'octets de sortie graphique> ::= empty_string
        | <image> <flux d'octets de sortie graphique>
        | <déclaration de sous-image> <flux d'octets de sortie graphique>
        | <opération d'angle de vue> <flux d'octets de sortie graphique>
        | <transmission control stt> <flux d'octets de sortie graphique>
<image> ::= <new picture sst> <program stt group> <end picture stt>
<déclaration de sous-image> ::= <subpicture header stt> <program stt
                    group><subpicture end stt>
<opération d'angle de vue> ::= <declare viewport stt>
        | <add subpicture to viewport stt>
        | <clear viewport stt>
<transmission control stt> ::= <set data length stt>
        | <extensive changes follow stt>
        | <end of extensive changes stt>
<program stt group> ::= empty_string | <program stt <program stt group>
<program stt> ::= <picture control stt> | <display stt> |
          <transmission control stt>
<picture control stt> ::= <escape to device stt>
        | <escape to highest coordinate system stt>
        | <restore coordinate system stt>
        | <mark stt>
```

```
        | <null stt>
        | <line mode stt>
        | <set intensity stt>
        | <subpicture declaration>
        | <simple instance stt>
        | <full instance stt>
        | <set character size stt>
<display stt> ::= <move absolute stt>
        | <move relative stt>
        | <draw absolute stt>
        | <draw relative stt>
        | <dot absolute stt>
        | <dot relative stt>
        | <move to mark and pop stt>
        | <draw to mark and pop stt>
        | <text and restore beam stt>
        | <text stt>
        | <text out stt>
<new picture stt> ::= ERASE
<end picture stt> ::= ENDPIC
<subpicture header stt> ::= SUBHED <identifier> count> <header info>
<header info> ::= 80-hex | 40-hex | C0-hex
<subpicture end stt> ::= SUBEND
<set viewport stt> ::= SETVW <viewport id> <x coordinate> <y coordinate> <x delta> <y delta>
<add subpicture to viewport stt> ::= AADSVW <identifier> <viewport id>
<clear viewport stt> ::= CLVW <viewport id>
<extensive changes follow stt> ::= DELAY
<end of extensive changes stt> ::= NODELAY
<escape to device stt> ::= ESCDEV <device code> <string>
<escape to highest coordinate system stt> ::= ESCTOP
<restore coordinate system stt> ::= RESLEV
<null stt> ::= NULL
<mark stt> ::= MARK
<line mode stt> ::= LINMOD <value>
<set character size stt> ::= SETCHS <x delta> <y delta>
<set data length stt> ::= SETDLN <value>
<move absolute stt> ::= MOVEA <x coordinate> <y coordinate>
<move relative stt> ::= MOVER <x delta> <y delta>
<draw absolute stt> ::= DRAWA <x coordinate> <y coordinate>
<draw relative stt> ::= DRAWR <x delta> <y delta>
<dot absolute stt> ::= DOTA <x coordinate> <y coordinate>
<dot relative stt> ::= DOTR <x delta> <y delta>
<move to mark and pop stt> ::= MOVEMK
<draw to mark and pop stt> ::= DRAWMK
<text and restore beam stt> ::= TEXTR <string>
<text stt> ::= TEXT <string>
<text out stt> ::= TEXTO <string>

<simple instance stt> ::= INST <identifier> <simple instance tail>
<full instance stt> ::= INSTF <identifier> <full instance tail>
<simple instance tail> ::= eight_bits_of_binary_0
            | <count> <tail code> <as clause> <at clause>
<tail code> ::= bit_pattern_indicating_what_clauses_follow
<full instance tail> ::= eight_bits_of_binary_0
            | <count> <tail code> <as clause> <at clause>
            <rotation clause> <portion clause>
            <uniform magnification clause>
            <separate magnification clause> <image size
            clause> <complete transformation clause>
<as clause> ::= empty_string | <identifier>
<at clause> ::= empty_string | <x coordinate> <y coordinate>
<rotation clause> ::= empty_string | <angle>
<portion clause> ::= empty_string | <x coordinate> <y coordinate> <x delta> <y delta>
```

<uniform magnification clause> ::= empty_string |floating point number>
<separate magnification clause> ::= empty_string | <floating point number> <floating point number>
<image size clause> ::= empty_string | <x delta> <y delta>
<complete transformation clause> ::= empty_string | six_<floating point number>'s

<angle> ::= 16-bit_non-negative_fractional_part_of_a_circle
<x coordinate> ::= <coordinate>
<y coordinate> ::= <coordinate>
<x delta> ::= <double coordinate>
<y delta> ::= <double coordinate>
<coordinate> ::= signed,_two_s-complement,_fraction_in_range -1/2_to_less_than_+1/2
<double coordinate> ::= signed,_two_s_complement,_fraction, range _strictly_between_-1_and_+1
<floating point number> ::= network_standard_floating_point
number_if_any
| 8-bit_two_s_complement_exponent_part and a
16-bit_two_s_complement_fraction_part <count>
::= 7-bit_non-negative_integer
| 15-bit_non-negative_integer_represented_in
"excess_2**15"_notation
<string> ::= <count> count_8-bit_bytes
<identifier> ::= <count> count_upper_case_letters_or_numbers
<viewport id> ::= <identifier>
<device code> ::= 8-bit_integer
<value> ::= 8-bit_integer

Appendice 2 Formules mathématiques pour les sous-images

Transformations

Dans le présent appendice, les positions dans un système de coordonnées logiques sont représentées par un vecteur ligne avec deux éléments, comme dans /_ x y_/. Les vecteurs et les matrices sont délimités par sortes de guillemets : /_ _/. Divers symboles sont utilisés pour représenter les paramètres dans une invocation de sous-image complète se rapportant à une transformation d'un système de coordonnées à un autre ; ils sont définis ci-dessous :

Mx et My : grossissements dans x et y à appliquer avant toute rotation. Les valeurs négatives indiquent la réflexion.
A : angle de rotation dans la gamme 0 à juste un peu moins que 2 π.
/_ |cx |cy_/ : centre (dans l'image invocante) de l'image de la sous-image.
|sx |sy : demi-tailles, dans les directions x et y, de l'image sur la page invocante en termes de système de coordonnées de la page invocante. Les valeurs négatives indiquent la réflexion.
/_ x y_/ : position sur la page invoquée.
/_ |x |y_/ : position sur la page invocante qui correspond à /_x y_/.
/_ Pcx Pcy_/ : centre de la portion du système de coordonnées de la sous-image invoquée qui est à transposer dans la page invocante. Elle prend la valeur par défaut de /_ 0 0_/ si elle n'est pas spécifiée.
Psx et Psy : demi-tailles, dans les directions x et y de la portion de la sous-image à transposer. Elles prennent toutes les deux la valeur +1/2 si elles ne sont pas spécifiées.

(Si un grossissement uniforme est spécifié, mettre Mx et My égal à la valeur spécifiée et procéder ci-dessous comme si ils étaient spécifiés.)

Si des grossissements sont spécifiés, on a ce qui suit :

/_ |x |y_/ = (/_ x y_/ - /_ Pcx Pcy_>) * / Mx/Psx 0 / * /_ 0 My/Psy_/

/ cos 0 sin 0 / * / 1/2 0 / + /_ |cx |cy_/ /_ -sin0 cos0_/ /_ 0 1/2_/

ou en d'autres termes,

1)
/_|x |y_/ = /_ x-Pcx y-Pcy_/ * / Mx cosA/2Psx Mx sinA/2Psx / /_ -My sinA/2Psy My cosA/2Psy_/ +/_ |cx |cy_/

(Le facteur de 1/2 est nécessaire parce que, par exemple, (x-Pcx)/Psx va de -1 à +1 pour les valeurs x dans la portion (c'est-à-dire, telles que |x-Pcx| <|Psx|) alors que l'image, dans le système de coordonnées de la sous-image invocante, ne devrait aller que de -1/2 à +1/2.)

Si la taille de l'image est spécifiée à la place du grossissement, nous avons ce qui suit :

/_ |x |y_/ = (/_ x y_/ - /_Pcx Pcy_/) * / 1/Psx 0 / * /_ 0 1/Psy _/

/ cosA sinA / * / |sx 0 / + /_ |cx |cy_/ /_ -sinA cosA _/ /_ 0 |sy _/

ou, en d'autres termes,

2)
/_|x |y_/ = /_x-Pcx y-Pcy_/ * /|sx cosA/Psx |sy sinA/Psx / /_-|sx sinA/Psy |sy cosA/Psy_/ + /_|cx |cy_/

En développant les quantités entre parenthèses dans les équations 1) et 2), nous avons :

3a) /_|x |y_/ = /_x y_/ * /Mx cosA/2Psx Mx sinA/2Psx / /_-My sinA/2Psy My cosA/2Psy_/

+ /_|cx-PcxMxcosA/2Psx+PcyMysinA/2Psy |cy-PcxMxsinA/2Psx-PcyMycosA/2Psy _/

et

3b) /_|x |y_/ = /_x y_/ * /|sx cosA/Psx |sy sinA/Psx / /_-|sx sinA/Psy |sy cosA/Psy_/

+ /_|cx-Pcx|sxcosA/Psx+Pcy|sxsinA/Psy |cy-Pcy|sysinA/Psx-Pcy|sycosA/Psy_/

Diverses substitutions intéressantes peuvent être faites en 3a) et 3b). Par exemple, si A=0 (pas de rotation), cela donne :

4a) /_|x |y_/ = /_x y_/ * /Mx/2Psx 0 / + /_|cx-PcxMx/2Psx |cy-PcyMy/2Psy_/ /_ 0 My/2Psy_/

4b) /_|x |y_/ = /_x y_/ * /|sx/Psx 0 / + /_|cx-Pcx|sx/Psx |cy-Pcy|sy/Psy_/ /_ 0 |sy/Psy_/

Autre exemple si il n'y a pas de découpage en portions (Pcx=Pcy=0, Psx=Psy=1/2):

5a) /_|x |y_/ = /_ x y_/ * /Mx cosA Mx sinA / + /_|cx |cy_/ /_-My sinA My sinA_/

5b) /_|x |y_/ = /_ x y_/ * /2|sx cosA 2|sy sinA / + /_|cx |cy_/ /_-2|sx sinA 2|sx cosA_/

Si de plus, 0=0, nous avons :

6a) /_|x |y_/ = /_ xMx+|cx yMy+|cy_/

6b) /_|x |y_/ = /_ x*2|sx+|cx y*2|sy+|cy_/

Bien sûr, dans tous les cas, la transformation de /_x y_/ en /_|x |y_/ peut être écrite sous la forme :

/_|x |y_/ = /_x y_/ * / 2 by 2 / + /_ translation _/ /_ matrix _/

En général, une transformation combinant une transformation linéaire et une translation est appelée une transformation affine.

Transformations avec niveaux incorporés.

La combinaison de deux transformations affines est encore une transformation affine. Bien sûr, si

/_|x |y_/ = /_x y_/ * / Mat1 / + /_ Tran1 _/ /_ _/
et
/_|x_ |y__/ = /_|x |y_/ * (/ Mat1 / * / Mat2 /) /_ _/ /_ _/ + (/_ Tran2 _/ + /_ Tran1 _/ * / Mat2 /) /_ _/

Et donc si on a des invocations de sous-image complètes incorporées, les données à tous les niveaux doivent être transformées uns seule fois, à savoir, par la transformation qui est la combinaison des transformations d'une seule étape à chaque niveau d'incorporation. Une nouvelle "grande combinaison" de transformation affine devrait être calculée chaque

fois qu'est invoquée une sous-image complète (après avoir achevé la transformation actuelle) en combinant la grande combinaison actuelle avec la transformation affine pour cette invocation de sous-image particulière.

Portions avec niveaux incorporés

Tant qu'aucune rotation n'est impliquée, ou même que les seules rotations sont de multiples de $\pi/2$, il est facile de mettre en œuvre des niveaux de portions. L'exposé des deux paragraphes suivants suppose qu'aucune rotation autre que de multiples entiers de $\pi/2$ n'est impliquée.

De la même façon qu'on peut garder la trace d'une transformation affine de "grande combinaison", on peut garder une grande combinaison de portions. À chaque niveau, on peut proceder comme suit : sauvegarder une copie de la grande portion actuelle, et utiliser l'inverse de la transformation affine de niveau unique (spécifiée dans l'invocation de la sous-image) pour déterminer quel rectangle de la page invoquée correspond à la grande portion actuelle (sur la page d'invocation).

Diverses relations peuvent exister entre ce rectangle et le rectangle spécifié (ou mentionné par défaut) dans l'invocation de sous-image. Elles peuvent être disjointes (auquel cas cette sous-image n'a pas besoin du tout d'être invoquéé) ; elles peuvent être égale (un cas facile) ; l'une peut contenir l'autre ou elles peuvent se chevaucher partiellement. Si il y a une intersection quelconque, cela sera un rectangle, et ce rectangle devient la nouvelle portion de grande combinaison.

Le problème avec les rotations autres que de multiples de $\pi/2$ est que l'image inverse du rectangle n'est plus dans l'orientation standard (bordures verticales et horizontales). Cela signifie que ses intersections avec la portion spécifiée dans l'invocation de sous-image peuvent avoir 3, 4, 5, 6, 7, ou 8 bordures (si elle n'est pas vide). Les niveaux plus profonds peuvent même devenir pires si ils impliquent aussi des rotations. Bien que travailler sur une telle situation ne présente pas de difficultés conceptuelles (pour certains) cela implique néanmoins une somme de calculs significativement plus élevée que dans le cas de simples bords horizontaux et verticaux.

Le présent protocole ne fait aucune recommandation dans le cas où des rotations autres que de multiples entiers de $\pi/2$ sont impliquées avec les portions. Il suggère que les portions incorporées soient traitées comme décrit ci-dessus dans le cas le plus direct.

[La présente RFC a été mise en forme pour entrée dans les archives en ligne des RFC par Helene Morin, Via Genie, 12/1999]