1- Qu'est-ce qu'une matrice?

Une matrice est une grille, chaque emplacement de la grille contenant une information. Par exemple, un échiquier est une matrice dans laquelle chaque carré contient une information spécifique: une pièce d'échec particulière ou l'absence d'une pièce d'échec.

Blanc vient de déplacer un pion de l'emplacement matriciel e2 à l'emplacement e4.

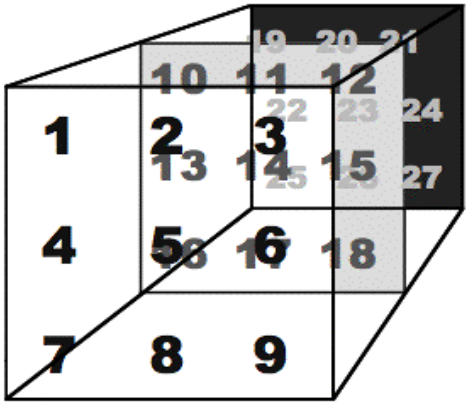
Cependant, pour les besoins de la discussion, supposons que les "informations" à chaque emplacement d'une matrice soient des données numériques (nombres). Voici une matrice avec un nombre à chaque emplacement de la grille.

	A	B	C	D
1	0.300	0.258	0.276	0.220
2	0.312	0.280	0.266	0.279
3	0.261	0.295	0.319	0.266

Un tableur est un exemple de matrice bidimensionnelle.

Nous appellerons chaque ligne horizontale de données une ***ligne*** et chaque ligne verticale de données une ***colonne***. Sur les cartes routières, sur les échiquiers ou les tableurs, les colonnes sont souvent identifiées par des lettres et les lignes par des chiffres. Cela nous permet de nous référencer à n'importe quel point de grille sur la carte en nous référant à sa colonne et à sa ligne. Dans les feuilles de calcul, un emplacement de la grille est appelé une cellule. Ainsi, dans l'exemple ci-dessus, la valeur numérique à la cellule C3 est 0,319.

Les deux images ci-dessus sont des exemples de matrices à deux dimensions: largeur (horizontale) et hauteur (verticale). Dans Jitter, une matrice peut avoir un nombre quelconque de dimensions de 1 à 32. (Une matrice unidimensionnelle est comparable à ce que les programmeurs appellent un ***tableau***.) Max possède déjà quelques objets qui sont utiles pour stocker des tableaux de nombres, tels que *table* et *multislider*. Il pourrait y avoir des cas, cependant, où une matrice unidimensionnelle dans Jitter serait plus utile.) Bien que ce soit un peu plus difficile à représenter sur papier, on peut certainement imaginer une matrice à trois dimensions, comme un cube ayant une largeur, une hauteur, et une profondeur. (Par exemple, une matrice de 3 cellules de large par 3 cellules de haut sur 3 cellules de profondeur aurait 3 x 3 x 3 = 27 cellules au total.)

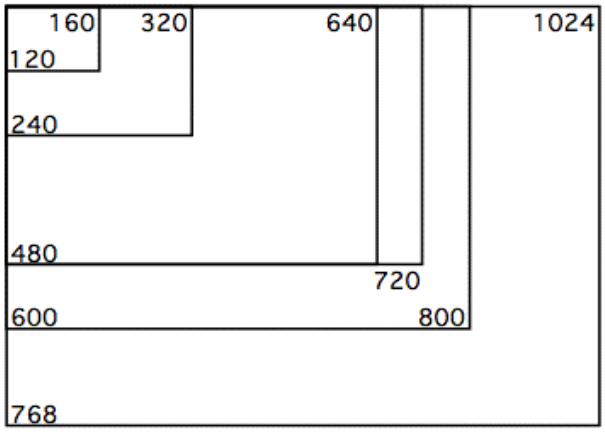


Une matrice 3x3x3 a 27 cellules.

Et bien que cela défie notre imagination visuelle et notre vocabulaire descriptif, on peut même avoir des matrices de quatre dimensions ou plus. Pour ce tutoriel, cependant, nous nous limiterons aux matrices à deux dimensions.

Un écran vidéo est un type de matrice

Un écran vidéo est composé de minuscules ***pixels*** individuels (éléments d'image), dont chacun affiche une couleur spécifique. Sur un écran d'ordinateur, la résolution de l'écran est généralement de l'ordre de 1024 pixels de large sur 768 pixels de haut, voire de 800x600 ou 640x480. Sur un écran de télévision (et dans la plupart des images vidéo conventionnelles), la résolution de l'écran est approximativement de 640x480, et sur les ordinateurs, elle est généralement traitée comme telle. Notez que dans tous ces cas, le rapport largeur / hauteur dit ***aspect ratio*** est 4:3. Dans le format DV plus large, le format de l'image est 3:2 et l'image est généralement de 720x480 pixels. La télévision haute définition (HDTV) spécifie encore un autre format d'image: 16:9. Dans ces tutoriels, nous travaillerons généralement avec un format d'image de 4:3, et le plus souvent avec des dimensions de pixel plus petites que la normale 320x240 ou même 160x120, juste pour économiser de l'espace dans le patch Max.



Tailles relatives de différentes dimensions standard de pixel

Une seule image de vidéo standard (c'est-à-dire une seule image vidéo à un moment donné) est composée de 640x480 = 307 200 pixels. Chaque pixel affiche une couleur. Afin de représenter numériquement la couleur de chaque pixel, avec une variété suffisante pour satisfaire nos yeux, nous avons besoin d'une très large gamme de valeurs de couleur différentes possibles.

Il existe de nombreuses manières différentes de représenter les couleurs numériquement. Une façon standard de décrire la couleur de chaque pixel dans les ordinateurs consiste à décomposer la couleur en ses trois composantes différentes - rouge, vert et bleu (alias RVB) - et en une composante supplémentaire de transparence/opacité (appelée canal ***alpha***). La plupart des programmes informatiques stockent donc la couleur d'un pixel unique sous la forme de quatre nombres distincts, représentant les composantes (ou canaux) alpha, rouge, vert et bleu . Ce schéma de représentation des couleurs à quatre canaux est communément appelé ***ARGB*** ou ***RGBA***, ***en fonction de la façon dont les pixels sont disposés dans la mémoire***.

Jitter ne fait pas exception à cet égard. Pour que chaque cellule d'une matrice représente un pixel de couleur, chaque cellule doit contenir quatre valeurs numériques (alpha, rouge, vert et bleu) et non une seule. Ainsi, une matrice qui stocke les données d'une image vidéo contiendra en réalité quatre valeurs dans chaque cellule.

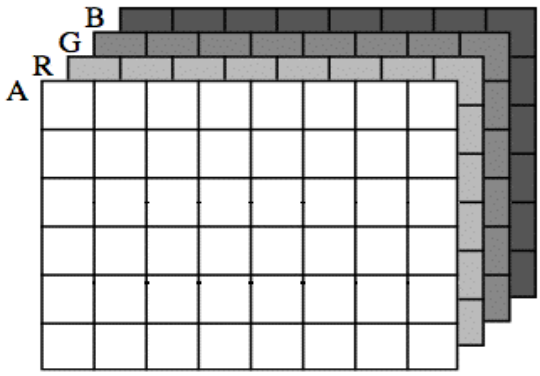
A: 255 R: 218 G: 111 B: 218	A: 255 R: 218 G: 111 B: 218	A: 254 R: 218 G: 112 B: 217	etc.
A: 255 R: 218 G: 111 B: 217	A: 255 R: 218 G: 111 B: 217	A: 254 R: 218 G: 112 B: 217	etc.
etc.	etc.	etc.	etc.

Chaque cellule d'une matrice peut contenir plus d'un nombre.

Une image vidéo est donc représentée dans Jitter comme une matrice à deux dimensions, chaque cellule représentant un pixel de l'image et chaque cellule contenant quatre valeurs représentant alpha, rouge, vert et bleu sur une échelle de 0 à 255. Afin de séparer ce concept de nombres multiples par cellule (ce qui est essentiel pour la vidéo numérique) du concept de dimensions dans une matrice, Jitter introduit l'idée de ***plans***.

Qu'est-ce qu'un plan?

Lors de l'allocation de mémoire pour les nombres dans une matrice, Jitter doit connaître l'étendue de chaque dimension (par exemple, 320 x 240) – et aussi le nombre de valeurs à contenir dans chaque cellule. Afin de garder la trace des différentes valeurs dans une cellule, Jitter utilise l'idée que chacune d'entre elles existe sur un plan séparé. Chacune des valeurs d'une cellule existe sur un plan particulier. On peut donc considérer qu'une image vidéo est une matrice à bidimensionnelle de quatre plans de données entrelacés.



Les valeurs de chaque cellule de cette matrice peuvent être considérées comme existant sur quatre plans virtuels.

En utilisant ce cadre conceptuel, nous pouvons traiter chaque plan (et donc chaque canal de l'information couleur) individuellement lorsque nous en avons besoin. Par exemple, si nous voulons augmenter la rougeur d'une image, nous pouvons simplement augmenter toutes les valeurs dans le plan rouge de la matrice et laisser les autres inchangées.

Le cas normal pour représenter la vidéo dans Jitter est d'avoir une matrice 2D avec quatre plans de données: alpha, rouge, vert et bleu. Les plans sont numérotés de 0 à 3, donc le canal alpha est dans le plan 0 et les canaux RGB dans les plans 1, 2 et 3.

Les données d'une matrice

Les ordinateurs ont différents formats internes pour stocker les nombres. Si nous savons quel type de nombre nous voulons stocker à un endroit particulier, nous pouvons économiser de la mémoire en n'allouant que l'espace mémoire dont nous avons réellement besoin pour chaque nombre. Par exemple, si nous voulons stocker les caractères alphabétiques occidentaux selon la norme de représentation ASCII, nous n'avons besoin que d'une plage de 0 à 255, ce qui signifie que nous n'avons besoin que de 8 bits d'espace mémoire pour stocker chaque caractère (car $2^8 = 256$ valeurs possibles différentes). Si nous voulons stocker une plus grande plage de nombres, nous pourrions utiliser 32 bits, ce qui nous donnerait des nombres entiers dans une plage allant de -2 147 483 648 à 2 147 483 647. Pour représenter des nombres comportant une partie décimale, comme 3.1416, nous utilisons ce que l'on appelle un système binaire à ***virgule flottante***, dans lequel certains des bits d'un nombre à 32 bits ou à 64 bits représentent la mantisse de la valeur et d'autres bits représentent l'exposant.

La plupart du temps, lorsque vous programmez dans Max (par exemple, si vous travaillez uniquement avec des données MIDI), vous n'avez peut-être pas besoin de savoir comment Max stocke les nombres. Cependant, lorsque vous programmez de l'audio numérique dans MSP, il est utile de savoir que MSP utilise des nombres à virgule flottante. (Vous rencontrerez des erreurs mathématiques si vous utilisez accidentellement le stockage de nombres entiers lorsque vous voulez stocker des fractions décimales.) Dans Jitter aussi, il est très utile d'être conscient des différents types de stockage de nombres utilisés par l'ordinateur, afin d'éviter d'éventuelles erreurs mathématiques.

Une matrice Jitter peut stocker des nombres à virgule flottante 64 bits (connus des programmeurs sous le nom de nombres float double précision ou double), des nombres à virgule flottante 32 bits (appelés simplement ***float***), des nombres entiers 32 bits (connus sous le nom de ***long int***, ou simplement ***int***), et des caractères 8 bits (appelés ***char***). Certains objets Jitter peuvent stocker leurs valeurs numériques dans un seul de ces formats possibles, vous n'avez donc pas à spécifier le type de stockage. Mais d'autres objets Jitter peuvent stocker leurs valeurs de différentes manières, le type de stockage doit donc être saisi comme argument dans l'objet, en utilisant les mots **char**, **long**, **float32** ou **float64**.

> *Concept important*: dans les cas où nous utilisons Jitter pour manipuler de la vidéo, la chose la plus importante à savoir sur le stockage des données dans les matrices Jitter est peut-être la suivante. Lorsqu'une matrice contient des données vidéo - comme dans les exemples des paragraphes précédents - elle suppose que les données sont représentées au format ARVB et que chaque cellule est donc susceptible de contenir des valeurs comprises entre 0 et 255 (souvent sur quatre plans). Pour cette raison, le type de stockage de données le plus courant est ***char***. Même si les valeurs stockées sont généralement numériques (et non des caractères alphabétiques), nous n'avons besoin que de 256 valeurs p différentes possibles pour chacune d'elles, de sorte que les 8 bits d'un ***char*** sont suffisants. Etant donné qu'une image vidéo contient ***un grand nombre*** de pixels et que chaque cellule peut contenir quatre

valeurs, il est logique que Jitter conserve de l'espace de stockage lorsqu'il traite un si grand nombre de valeurs. La manipulation des données vidéo étant l'activité principale de nombreux objets Jitter, la plupart des objets matriciels utilisent le type de stockage ***char*** par défaut. Pour les images ou les vidéos monochromes (niveaux de gris), un seul plan de données char est suffisant.