COURS 2 - TYPES, VARIABLES, OPÉRATEURS

G. Bianchi, G. Blin, A. Bugeau, S. Gueorguieva, R. Uricaru

2015-2016

Programmation 1 - uf-info.ue.prog1@diff.u-bordeaux.fr

université de BORDEAUX

BILAN DE L'EXERCICE À LA MAISON

ANALYSE GLOBALE

▷ Pourquoi des cases de même taille (unité indivisible) ?

  ▷ Intérêts ?

  ▷ Défauts ?

▷ Limitation forte des données enregistrées dans cet exercice ?

  ▷ Comment surmonter cette limitation ?

ANALYSE GLOBALE

▷ Pourquoi des cases de même taille (unité indivisible) ?

▷ Intérêts ?

Plus simple à construire et à parcourir

▷ Défauts ?

▷ Limitation forte des données enregistrées dans cet exercice ?

▷ Comment surmonter cette limitation ?

ANALYSE GLOBALE

▷ Pourquoi des cases de même taille (unité indivisible) ?

▷ Intérêts ?

Plus simple à construire et à parcourir

▷ Défauts ?

Choix de l'unité et éventuelle perte de place si mal choisie

▷ Limitation forte des données enregistrées dans cet exercice ?

▷ Comment surmonter cette limitation ?

ANALYSE GLOBALE

▷ Pourquoi des cases de même taille (unité indivisible) ?

▷ Intérêts ?

Plus simple à construire et à parcourir

▷ Défauts ?

Choix de l'unité et éventuelle perte de place si mal choisie

▷ Limitation forte des données enregistrées dans cet exercice ?

▷ Comment surmonter cette limitation ?

Un seul type (ici des entiers). Il faudrait être capable de coder d'autre types de données et d'indiquer qui est quoi

...

ANALYSE DE QUELQUES CAS

- Utilisation des couleurs
  - 1 couleur = 1 chiffre (e.g., 1 = bleu, 2 = rouge, 3 = noir, ...)
  - 1 ligne pour le codage des chiffres
  - 1 couleur = 1 valeur (e.g., vert = 2015)
  - Codage binaire (uniquement 2 couleurs)
  - 1 couleur = 1 type d'information (e.g., rouge = Année de naissance)

ANALYSE DE QUELQUES CAS

▷ Utilisation des couleurs

▷ 1 couleur = 1 chiffre (e.g., 1 = bleu, 2 = rouge, 3 = noir, ...)

▷ 1 ligne pour le codage des chiffres

▷ 1 couleur = 1 valeur (e.g., vert = 2015) ⚠

▷ Codage binaire (uniquement 2 couleurs)

▷ 1 couleur = 1 type d'information (e.g., rouge = Année de naissance) ⚠

ANALYSE DE QUELQUES CAS

- Mémoire
  - Cases blanches = pas d'information
  - 1 information par ligne
  - Séparateur de données
  - Sens lecture (de G à D, de D à G, de H à B, ...)

ANALYSE DE QUELQUES CAS

- Informations de décodage
  - Association couleurs
  - Ordre de stockage
  - Localisation
  - Longueur des données

ANALYSE DE QUELQUES CAS

- Différences
  - Sens de lecture
  - Choix codage
  - Organisation mémoire
  - Longueur de données

ANALYSE DE QUELQUES CAS

Les réponses de L'Eponge Bob

Fiche signalétique

Age

16

Année de naissance

1999

Annee d'obtention de la L1, du BTS ou du DUT

2002

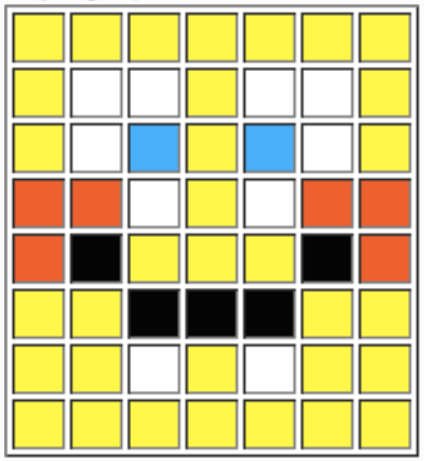
Département de naissance

-50000 Bikini Bottom

Nombre de voyelles (a, e, i, o, u ou y) dans votre nom

3

Mémoire



ANALYSE DE QUELQUES CAS

- Une couleur est associée à un chiffre
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - Sans le codage, impossible de décoder
- Les nombres sont représentés en représentation positionnelle (i.e., unité, dizaine, centaine, ...) comme à l'école
  -  = 2*1000 + 0*100 + 1*10 + 5*1
- Une information par ligne
  - Problème ?
- La taille a priori de certains nombres n'est pas connue
  - Solution ?

ANALYSE DE QUELQUES CAS

- Une couleur est associée à un chiffre
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - Sans le codage, impossible de décoder
- Les nombres sont représentés en représentation positionnelle (i.e., unité, dizaine, centaine, ...) comme à l'école
  -  = 2*1000 + 0*100 + 1*10 + 5*1
- Une information par ligne
  - Problème ? Perte de place
- La taille a priori de certains nombres n'est pas connue
  - Solution ?

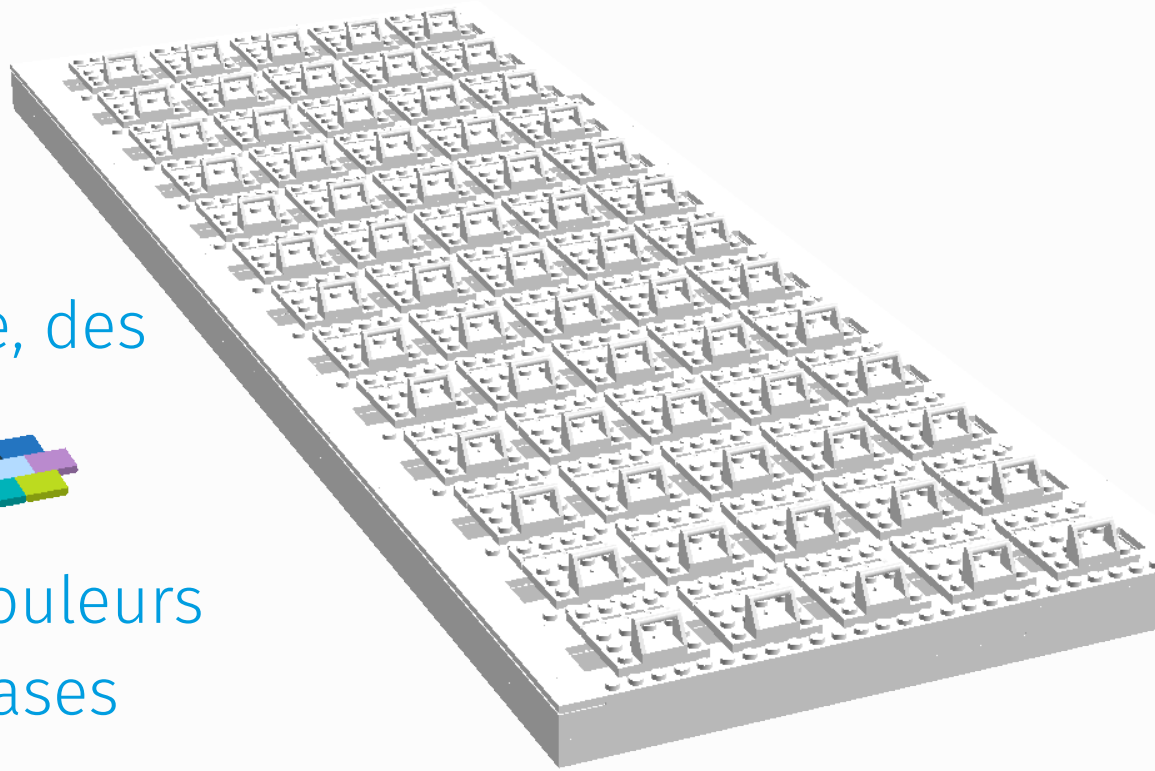
ANALYSE DE QUELQUES CAS

- Une couleur est associée à un chiffre
  - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  - Sans le codage, impossible de décoder
- Les nombres sont représentés en représentation positionnelle (i.e., unité, dizaine, centaine, ...) comme à l'école
  -  = 2*1000 + 0*100 + 1*10 + 5*1
- Une information par ligne
  - Problème ? Perte de place
- La taille a priori de certains nombres n'est pas connue
  - Solution ? Fixer une taille maximale

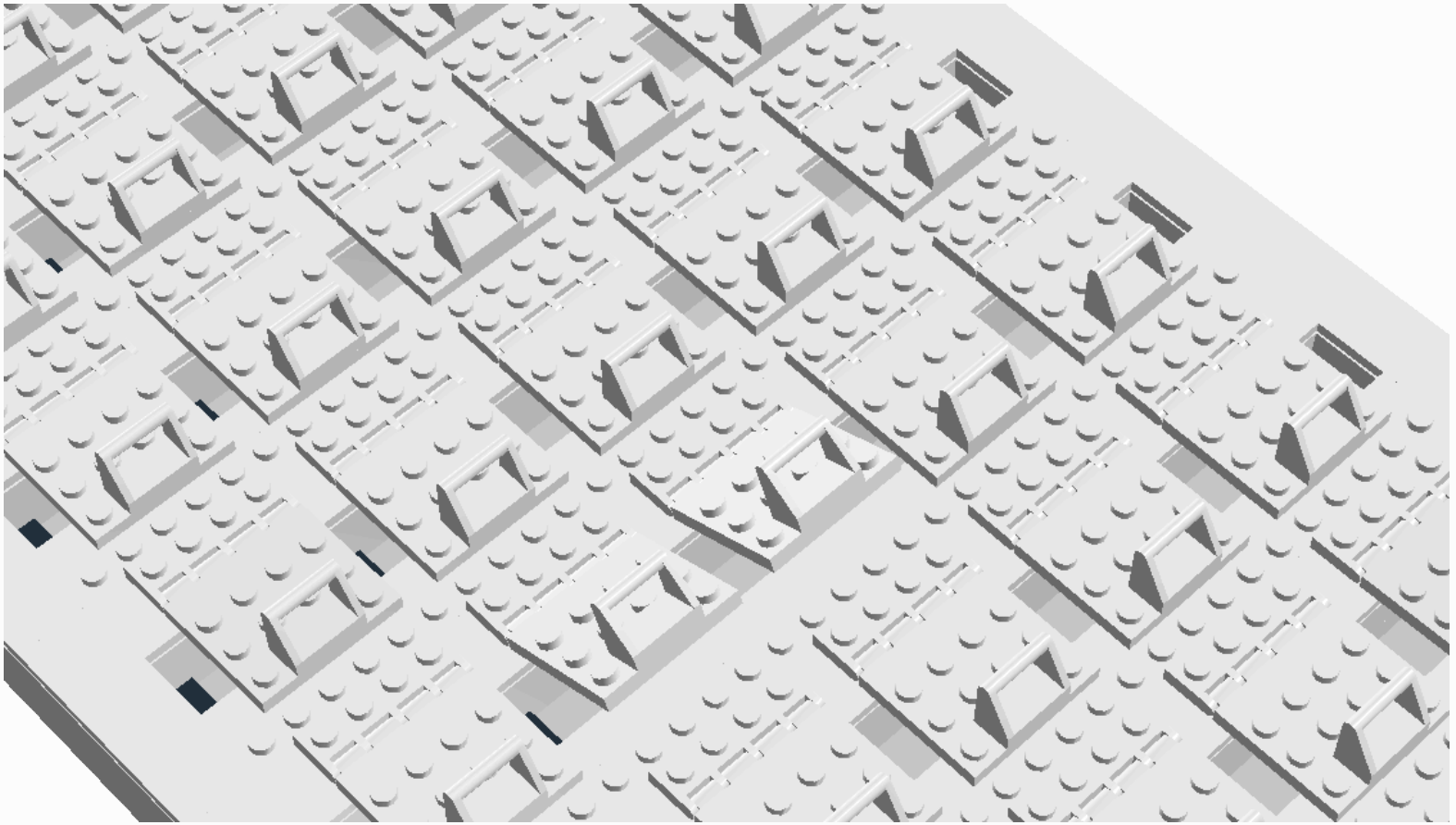
UN PEU DE LEGO

MODÈLE PLUS ÉVOLUÉ

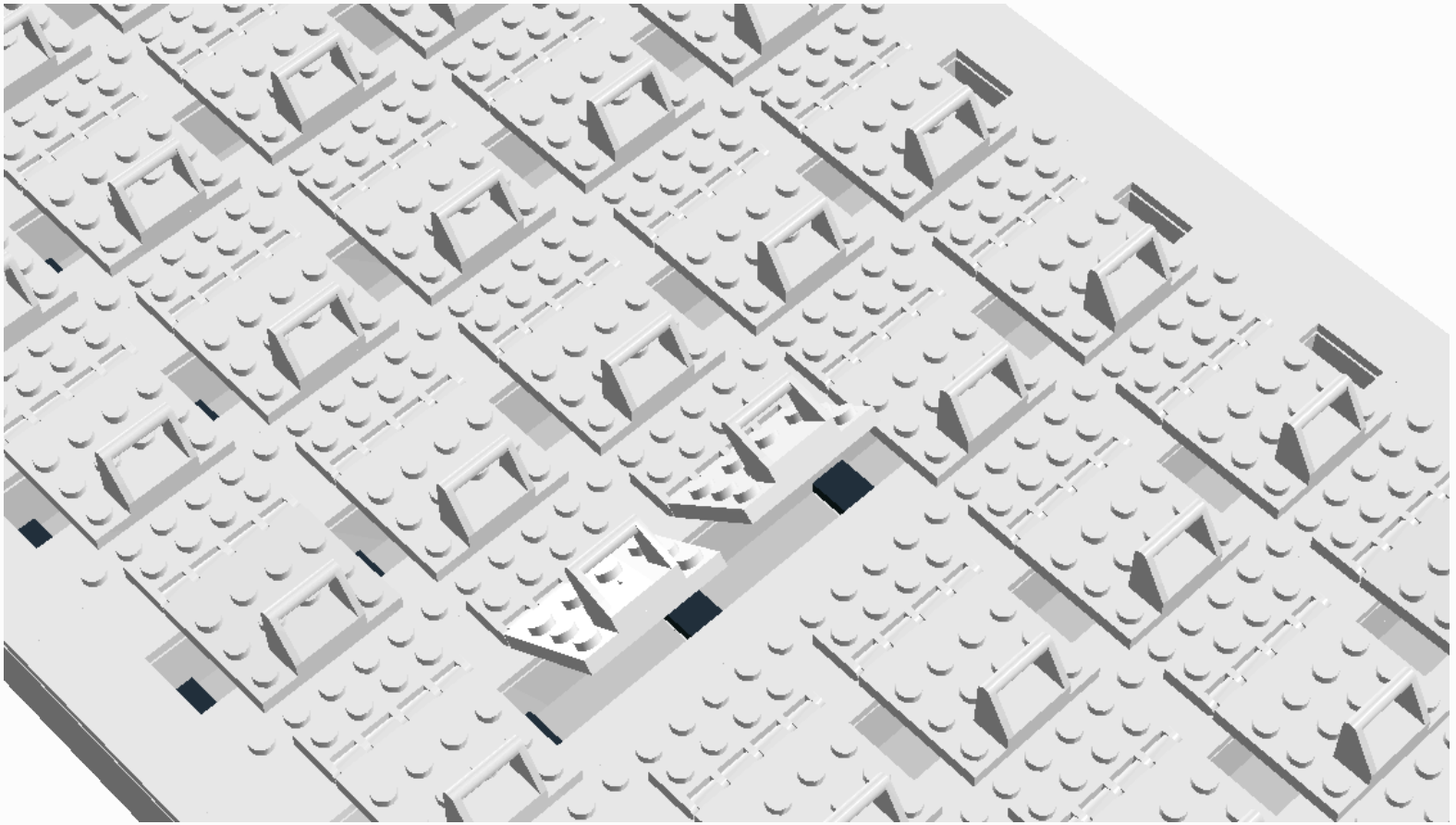
- Objectif :
  - stocker du texte, des entiers et réels
- Moyens :
  - seulement 10 couleurs
  - un bloc de 65 cases



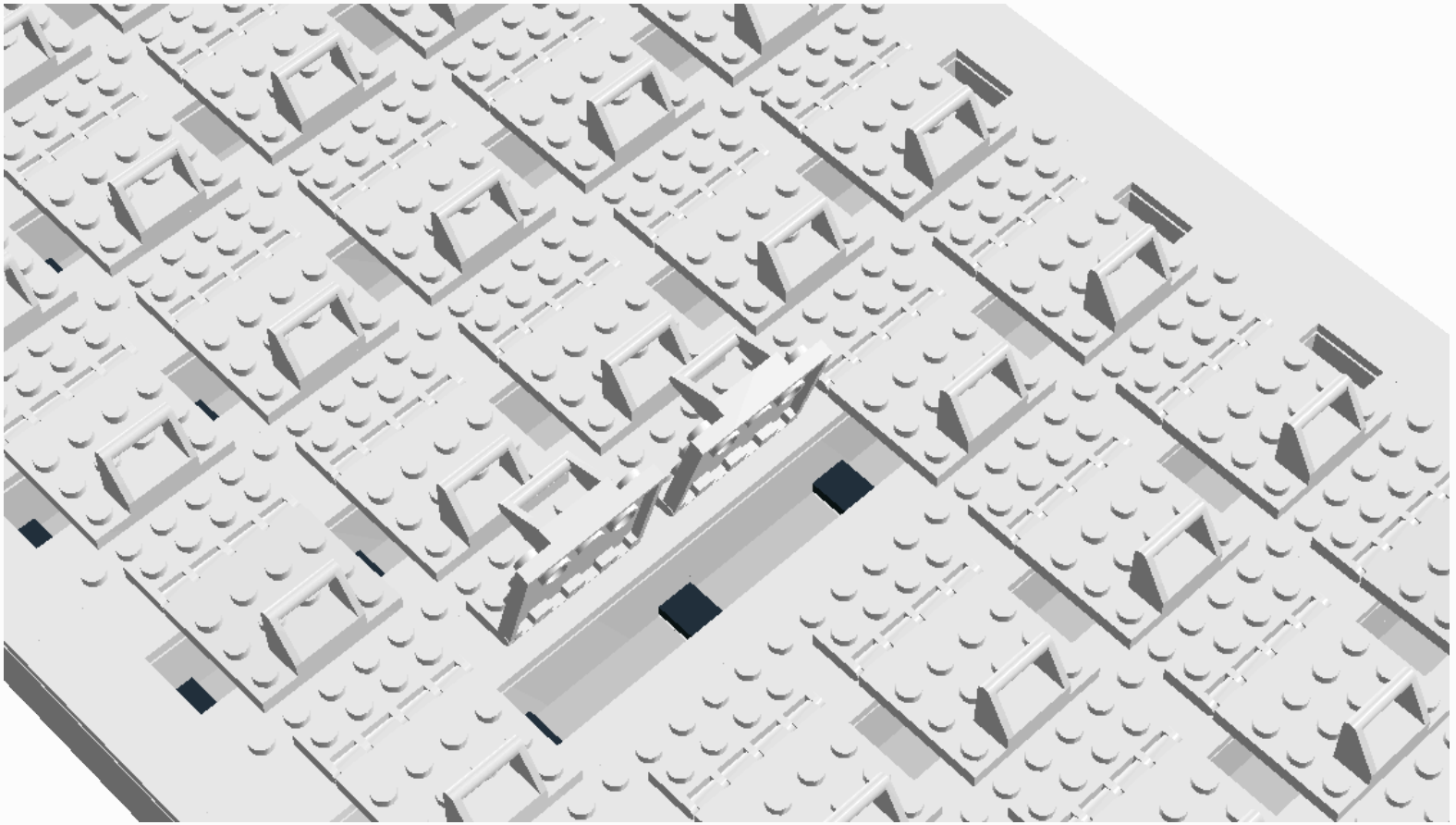
MODÈLE PLUS ÉVOLUÉ



MODÈLE PLUS ÉVOLUÉ



MODÈLE PLUS ÉVOLUÉ



Système d'adressage

▷ Proposer un système de coloriage pour que chaque case puisse être identifiée.

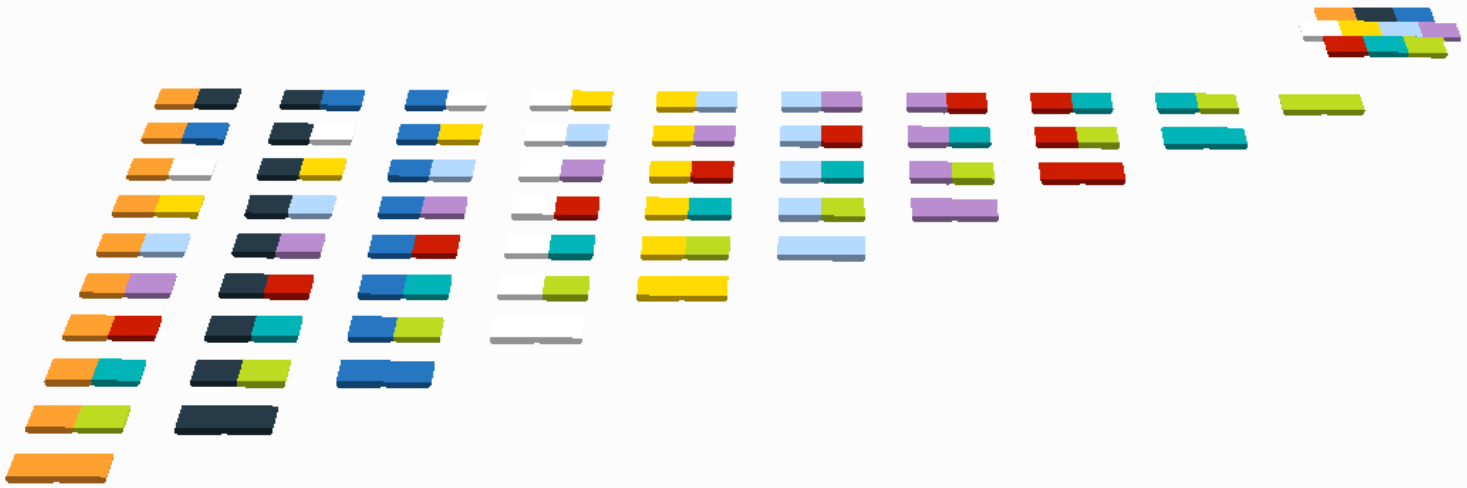
▷ Rappel : 65 cases et 10 couleurs...

Système d'adressage

▷ Proposer un système de coloriage pour que chaque case puisse être identifiée.

▷ Rappel : 65 cases et 10 couleurs...

▷ Il faut au moins des paires de couleurs par case



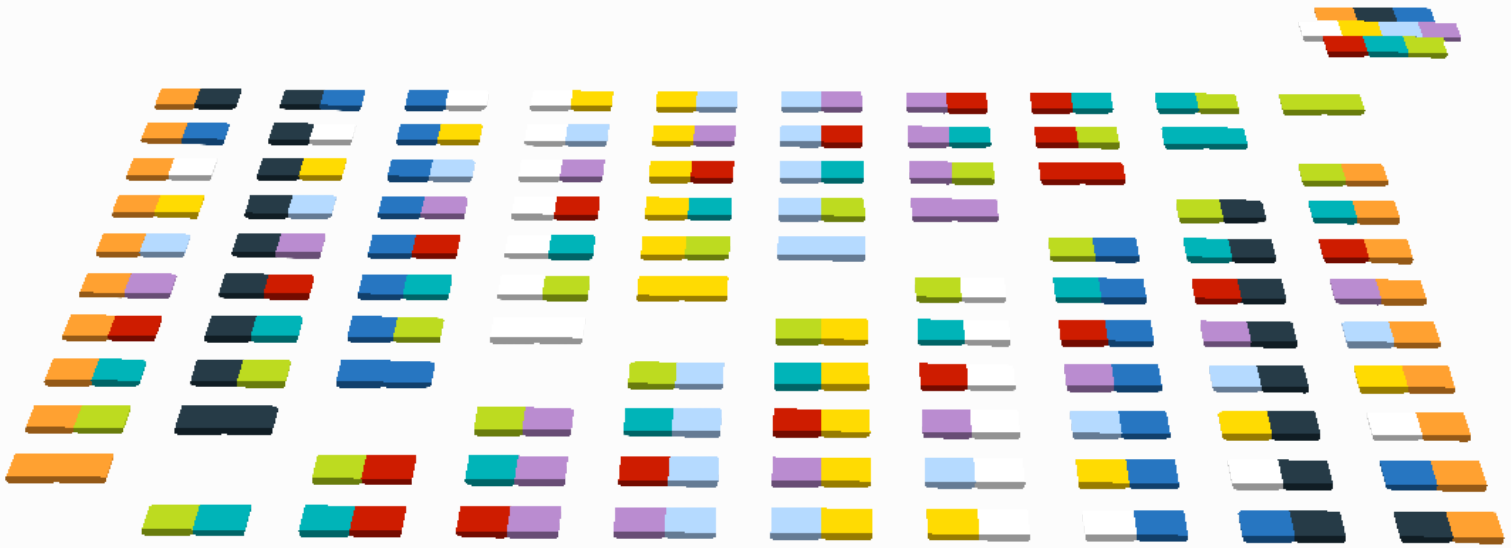
$$\frac{n(n+1)}{2} = 55$$

SYSTÈME D'ADRESSAGE

▷ Proposer un système de coloriage pour que chaque case puisse être identifiée.

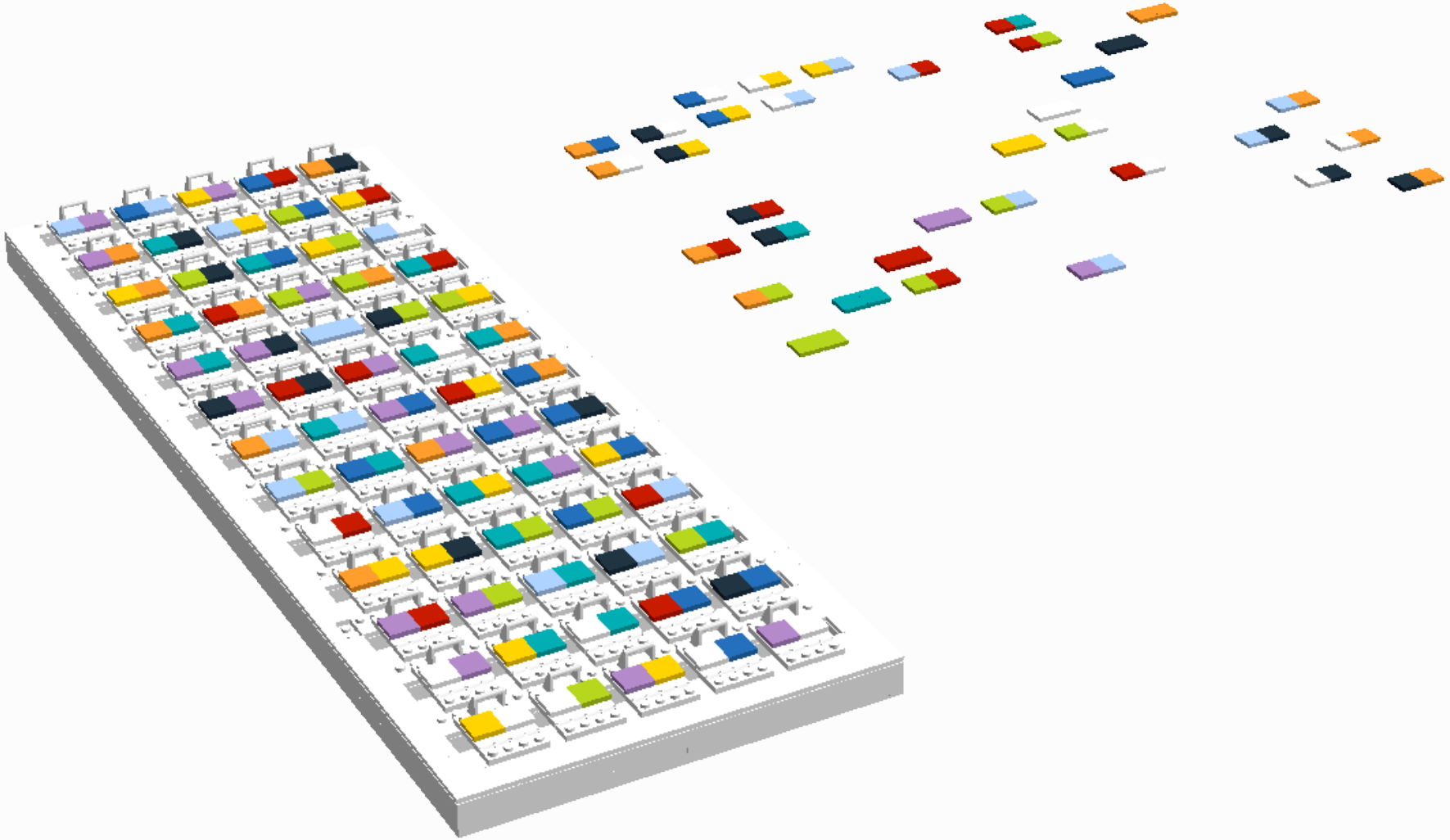
▷ Rappel : 65 cases et 10 couleurs...

▷ Il faut au moins des paires de couleurs par case et ordonnées



SYSTÈME D'ADRESSAGE

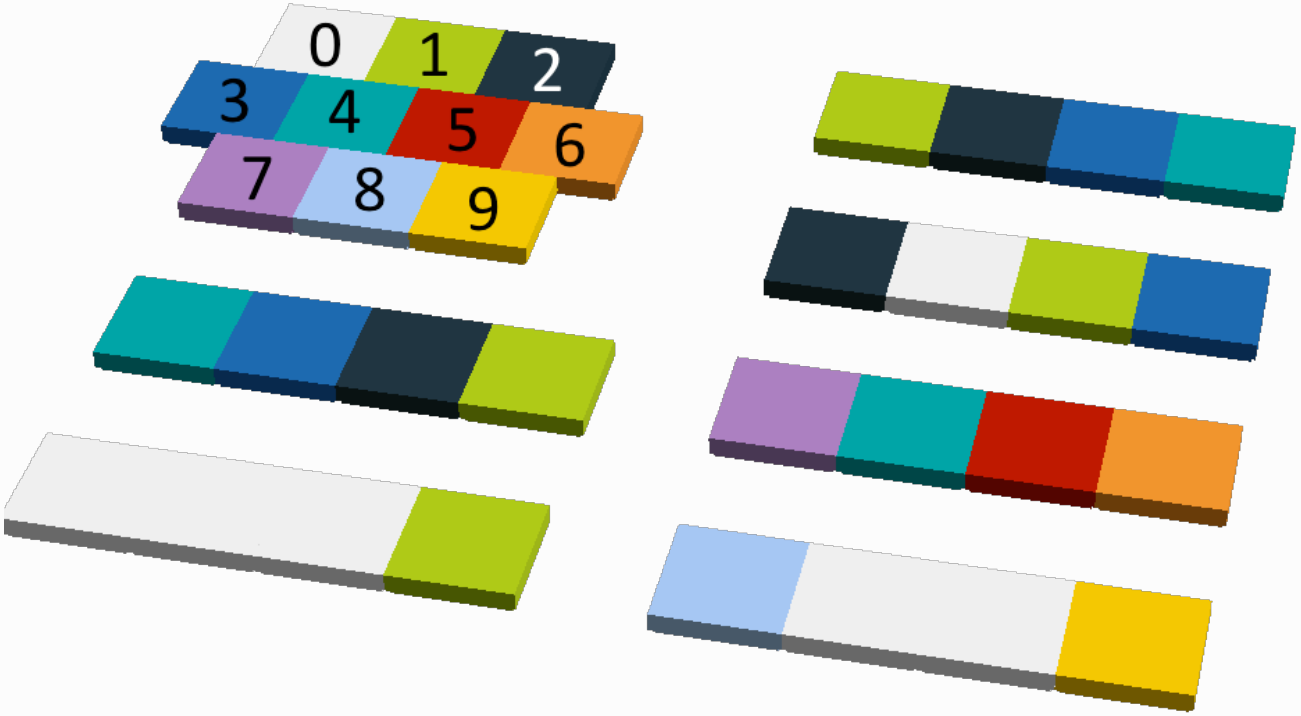
▷ Proposer un système de coloriage pour que chaque case puisse être identifiée.



Stockage d'entiers positifs de 0 à 9999

▷ Représentation positionelle

▷ $123 = 1 * 100 + 2 * 10 + 3 * 1$



▷ Notons qu'une limite en nombre de chiffres est nécessaire

Stockage d'entiers négatifs

- Peut-on stocker des entiers négatifs avec ce principe ?
  - Comment représenter le signe ?
  - Limitations induites ?

Stockage d'entiers négatifs

- ▷ Peut-on stocker des entiers négatifs avec ce principe ?
  - ▷ Comment représenter le signe ?

"Sacrifier" une des positions (e.g., la première)

  - ▷ Limitations induites ?

Stockage d'entiers négatifs

▷ Peut-on stocker des entiers négatifs avec ce principe ?

▷ Comment représenter le signe ?

"Sacrifier" une des positions (e.g., la première)

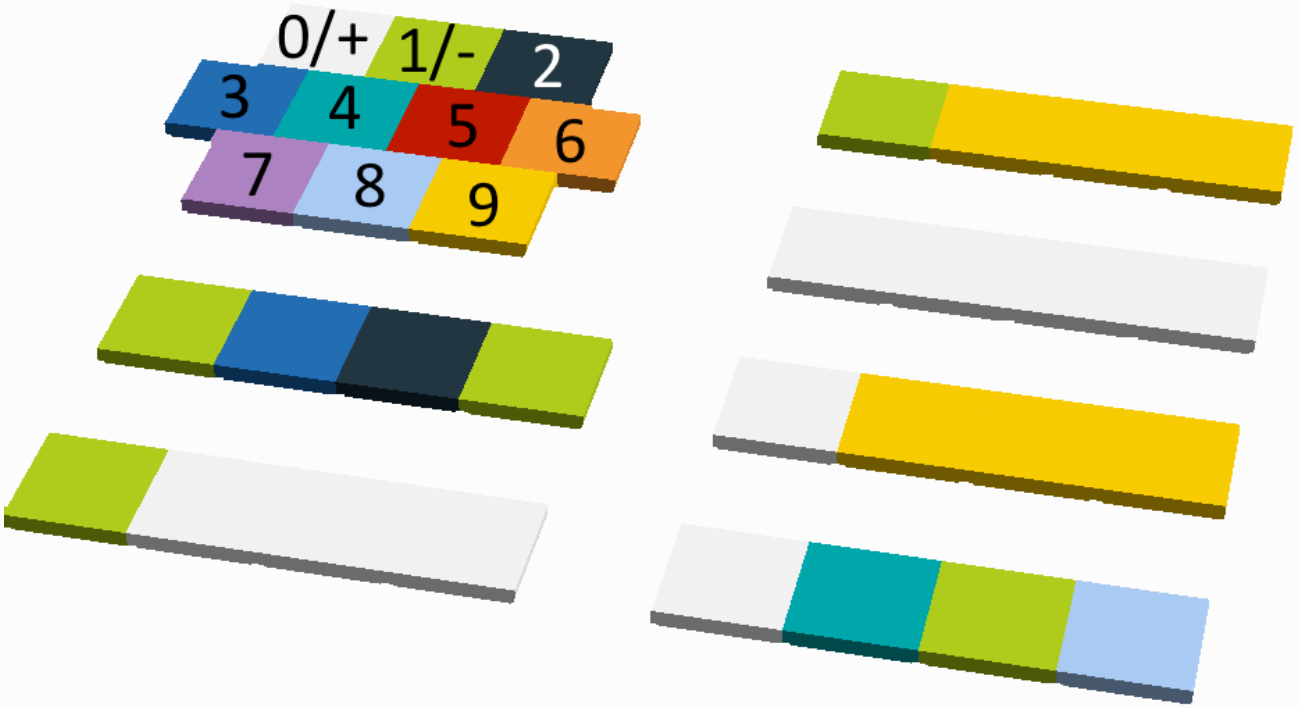
▷ Limitations induites ?

Des nombres plus petits représentables

▷ Y en a t-il moins de représentables pour autant ?

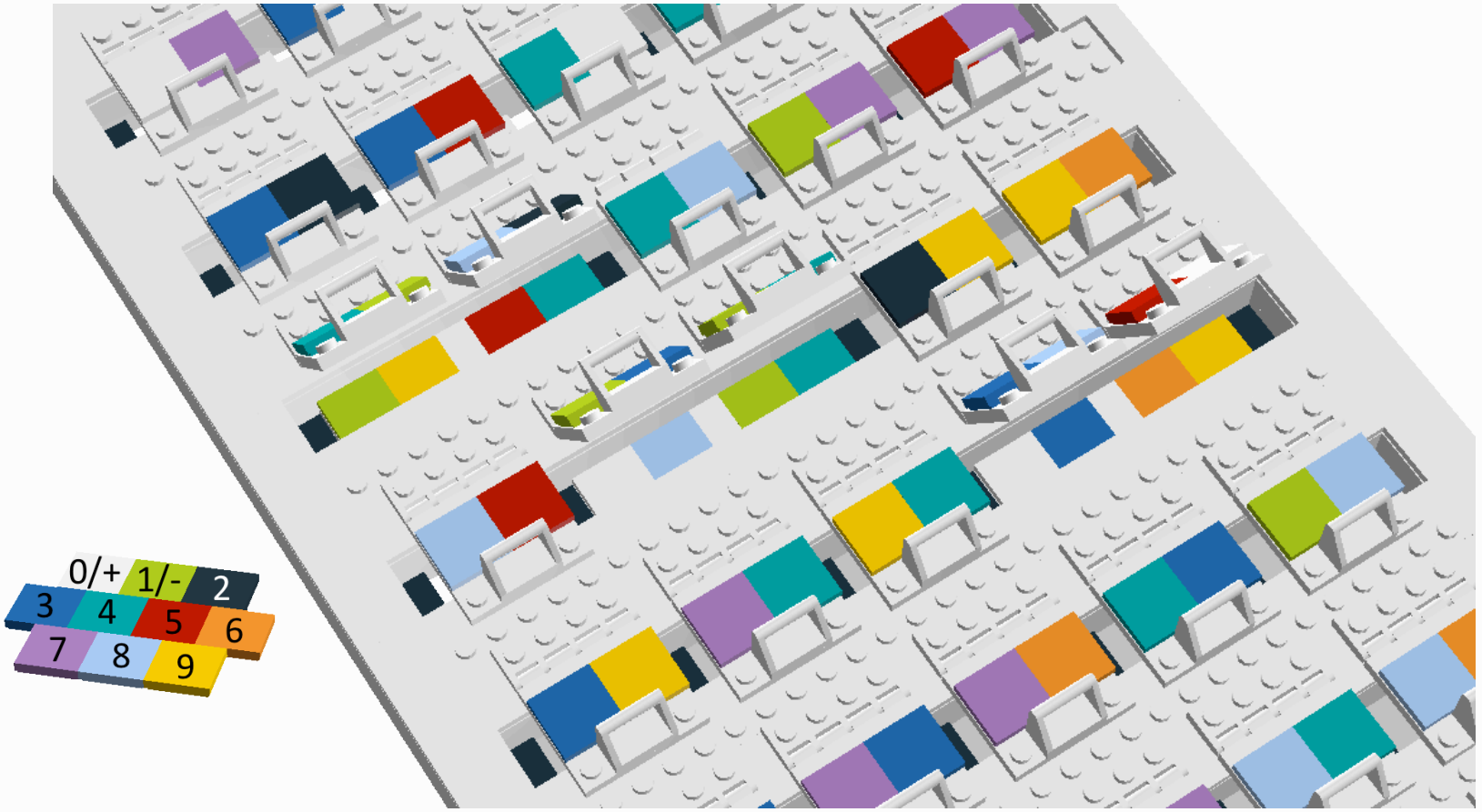
Stockage d'entiers négatifs

▷ Peut-on stocker des entiers négatifs avec ce principe ?



▷ Entiers entre -999 et +999 avec deux codages de 0

EXEMPLE DE STOCKAGE D'ENTIERS



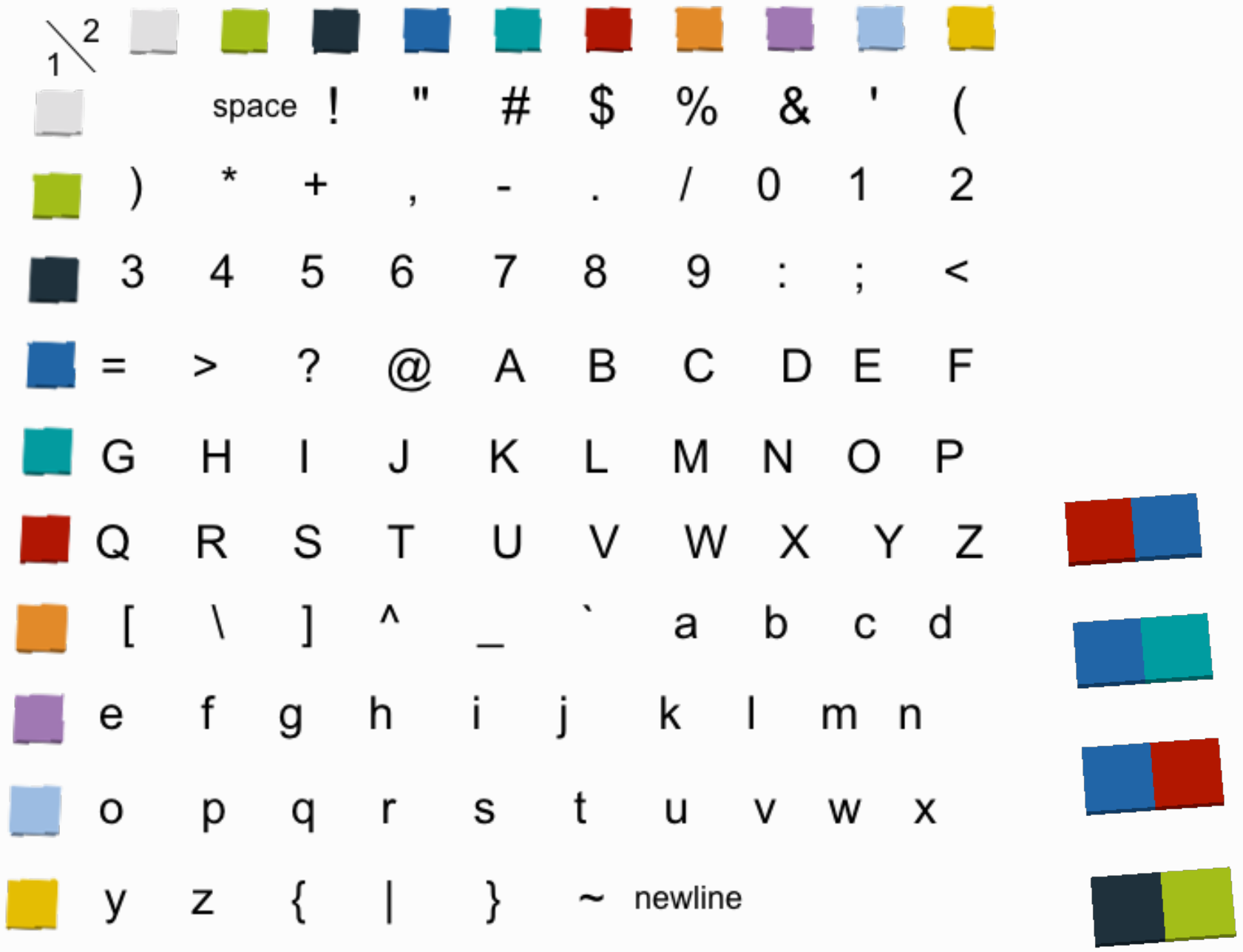
STOCKAGE DE TEXTE

- Comment stocker du texte en anglais ?
  - Base de 96 caractères

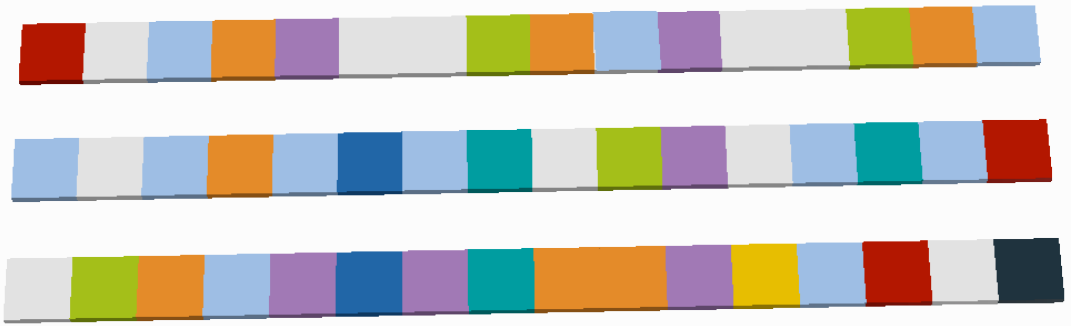
```
space ! ” # $ % & ’ ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ? @ A
B C D E F G H I J K L M N O P Q R S
T U V W X Y Z [ \ ] ^ _ ‘ a b c d e
f g h i j k l m n o p q r s t u v w
x y z { | } ~ newline
```

- Comment stocker un caractère ?

CODAGE D'UN CARACTÈRE

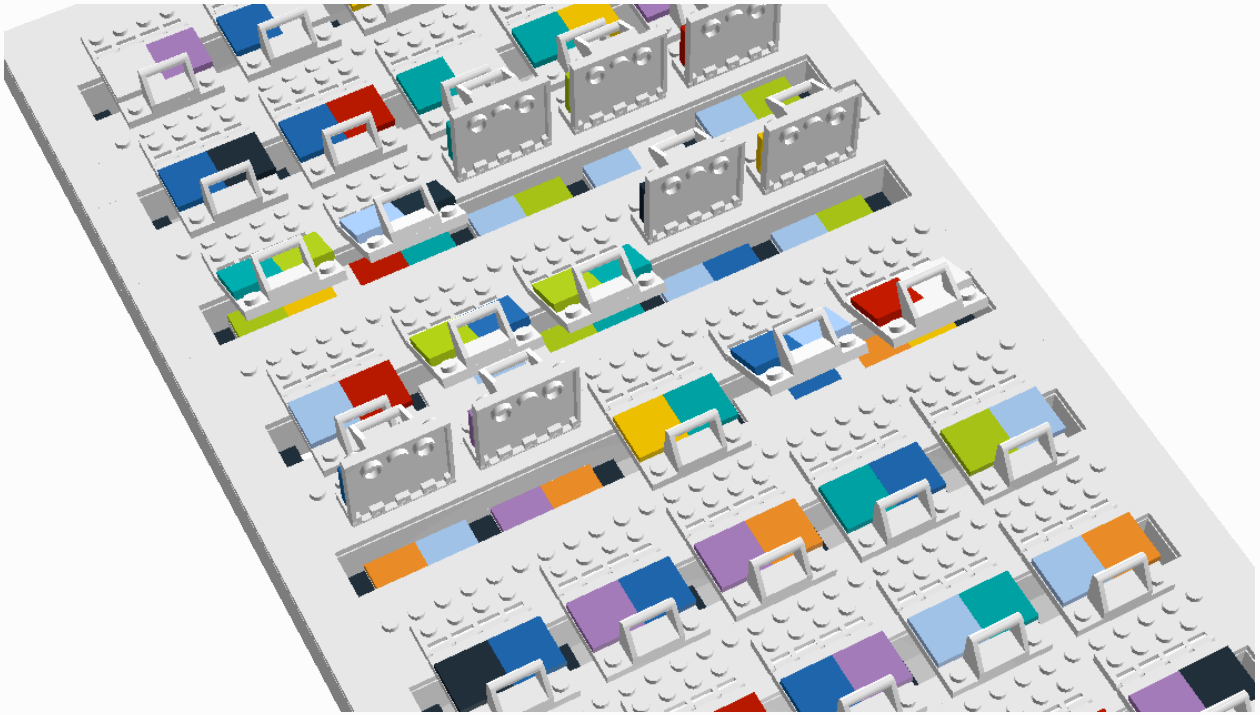


CODAGE D'UNE CHAÎNE DE CARACTÈRES

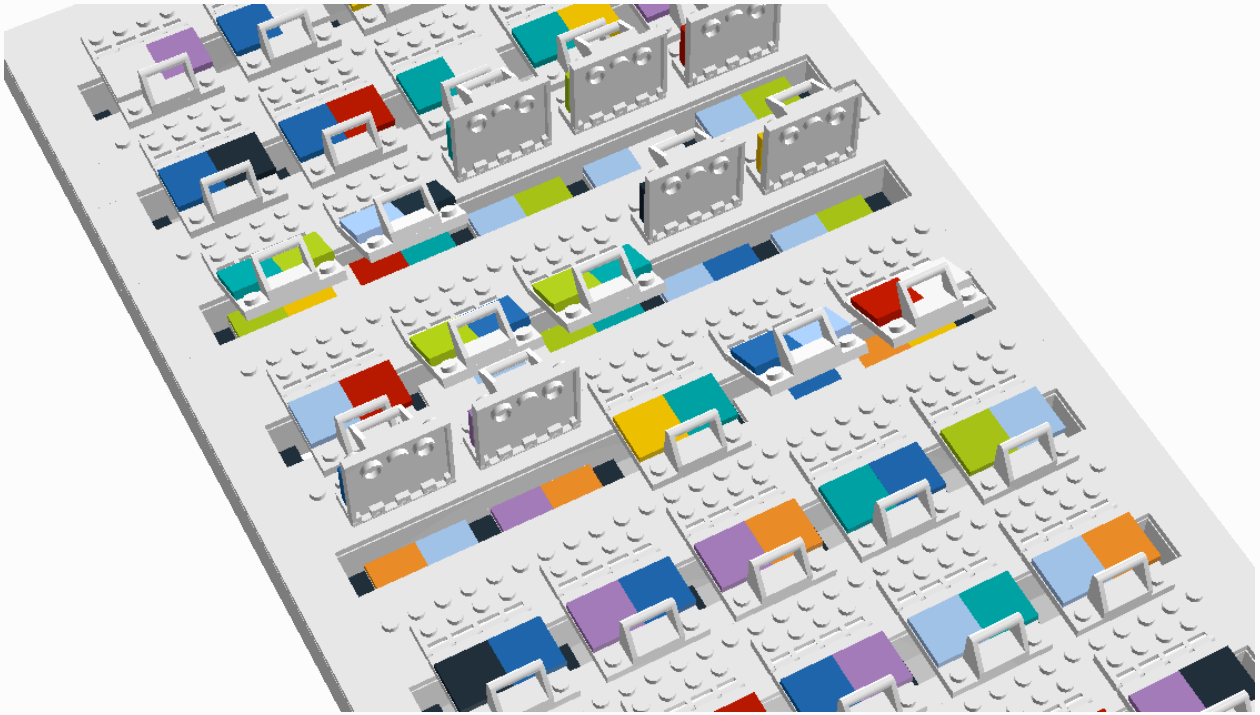


1 \ 2										
	space	!	"	#	$	%	&	'	(	
)	*	+	,	-	.	/	0	1	2	
3	4	5	6	7	8	9	:	;	<	
=	>	?	@	A	B	C	D	E	F	
G	H	I	J	K	L	M	N	O	P	
Q	R	S	T	U	V	W	X	Y	Z	
[	\	]	^	_	`	a	b	c	d	
e	f	g	h	i	j	k	l	m	n	
o	p	q	r	s	t	u	v	w	x	
y	z	{	|	}	~	newline				

PROBLÉMATIQUE DU STOCKAGE DES CHAÎNES DE CARACTÈRES



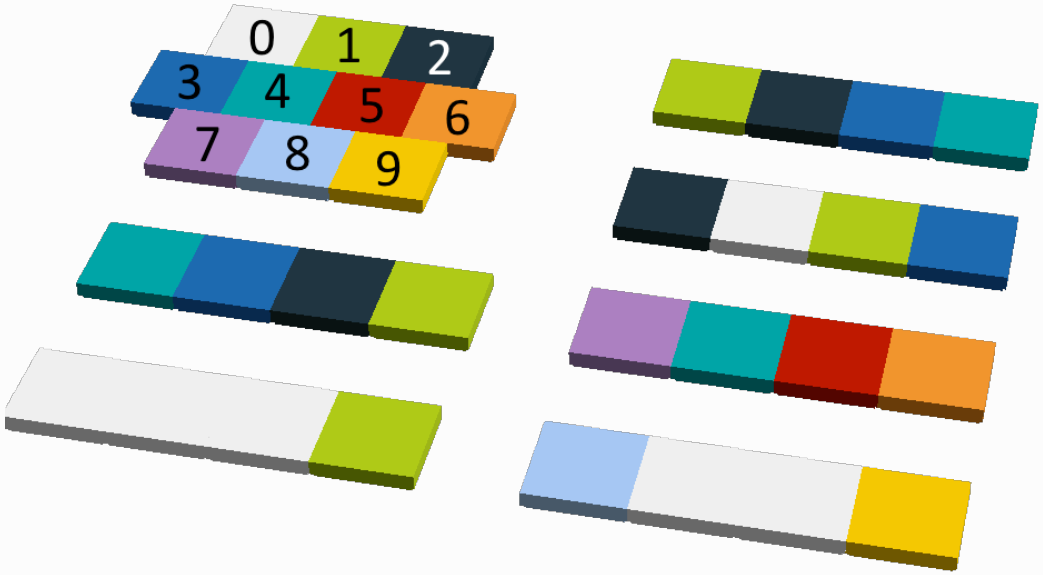
PROBLÉMATIQUE DU STOCKAGE DES CHAÎNES DE CARACTÈRES



- Convention de fin de chaîne de caractères
- Nécessité de consécutivité dans la mémoire

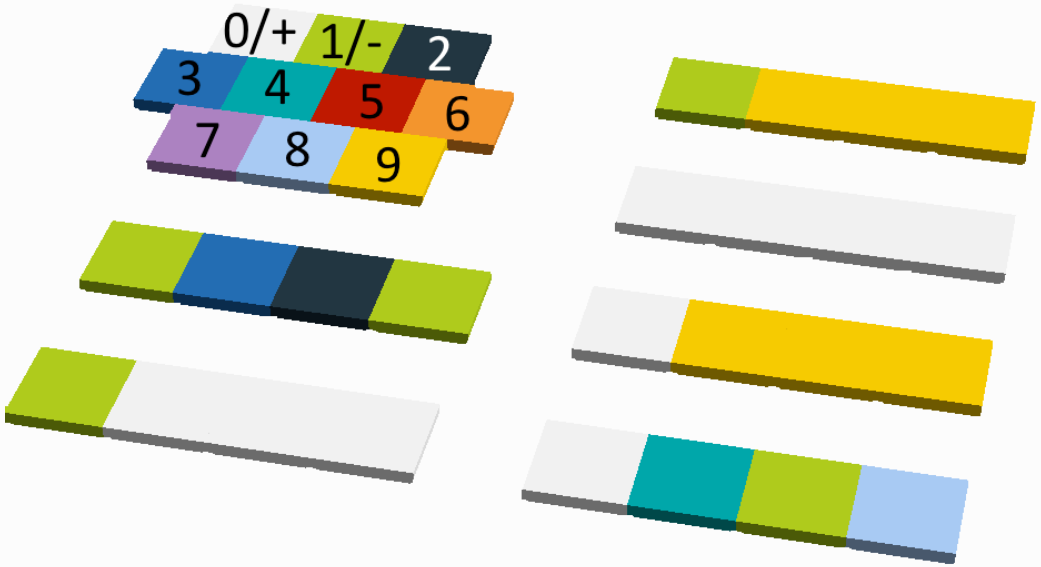
STOCKAGE DES RÉELS POSITIFS - REP. À VIRGULE FIXE

- Avec 4 couleurs
  - de 000,0 à 999,9 par pas de 0,1
  - de 00,00 à 99,99 par pas de 0,01
  - de 0,000 à 9,999 par pas de 0,001
  - de 0,000 à 0,9999 par pas de 0,0001



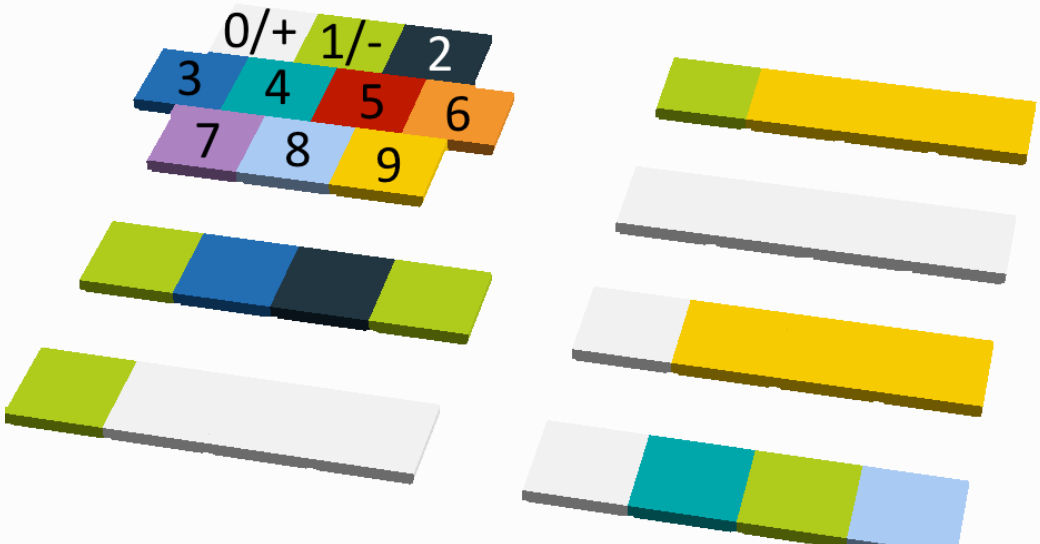
STOCKAGE DES RÉELS POSITIFS - REP. À VIRGULE FIXE

- Avec 4 couleurs
  - de ~~000,0~~ -99,9 à ~~999,9~~ 99,9 par pas de 0,1
  - de ~~00,00~~ -9,99 à ~~99,99~~ 9,99 par pas de 0,01
  - de ~~0,000~~ -0,999 à ~~9,999~~ 0,999 par pas de 0,001
  - de ~~0,000 à 0,9999~~ par pas de 0,0001



STOCKAGE DES RÉELS - REP. À VIRGULE FLOTTANTE

- 1 couleur = position de la virgule (e.g., à droite du signe)
- Avec 4 couleurs:
  - de -99 à 99 par pas de 1 [pos(,) = 0]
  - de -9,9 à +9,9 par pas de 0,1 [pos(,) = 1]
  - de -0,99 à +0,99 par pas de 0,01 [pos(,) = 2]
- Quel est l'intervalle avec pos(,) = 3 ?
- Plus de cases ?



STOCKAGE DES RÉELS - REP. À VIRGULE FLOTTANTE

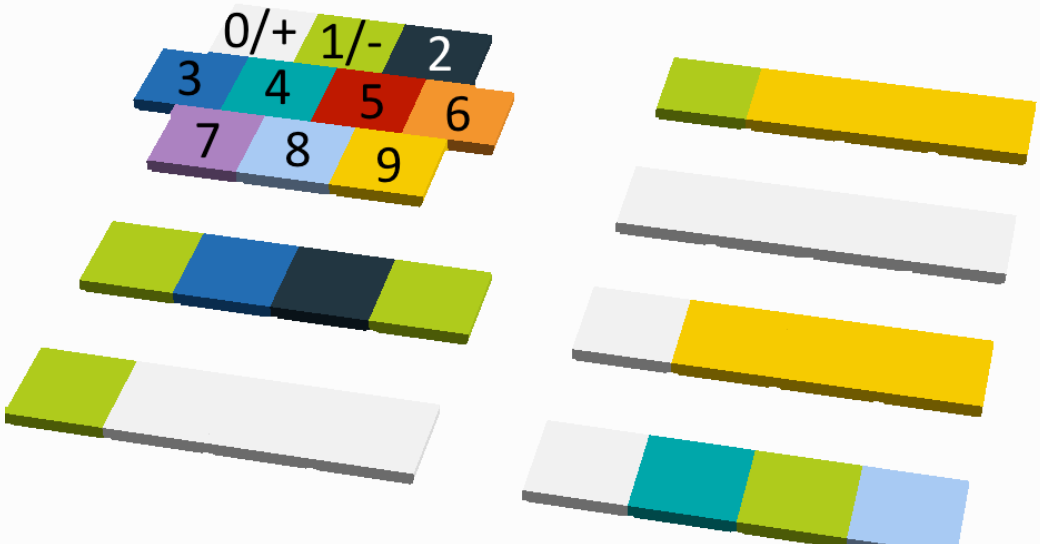
- ▷ 1 couleur = position de la virgule (e.g., à droite du signe)
- ▷ Avec 4 couleurs:
  - ▷ de -99 à 99 par pas de 1 [pos(,) = 0]
  - ▷ de -9,9 à +9,9 par pas de 0,1 [pos(,) = 1]
  - ▷ de -0,99 à +0,99 par pas de 0,01 [pos(,) = 2]
- ▷ Quel est l'intervalle avec pos(,) = 3 ?

-0,099 à -0,001

0

+0,001 à +0,099

- ▷ Plus de cases ?



STOCKAGE DES RÉELS - REP. À VIRGULE FLOTTANTE

- ▷ 1 couleur = position de la virgule (e.g., à droite du signe)
- ▷ Avec 4 couleurs:
  - ▷ de -99 à 99 par pas de 1 [pos(,) = 0]
  - ▷ de -9,9 à +9,9 par pas de 0,1 [pos(,) = 1]
  - ▷ de -0,99 à +0,99 par pas de 0,01 [pos(,) = 2]
- ▷ Quel est l'intervalle avec pos(,) = 3 ?

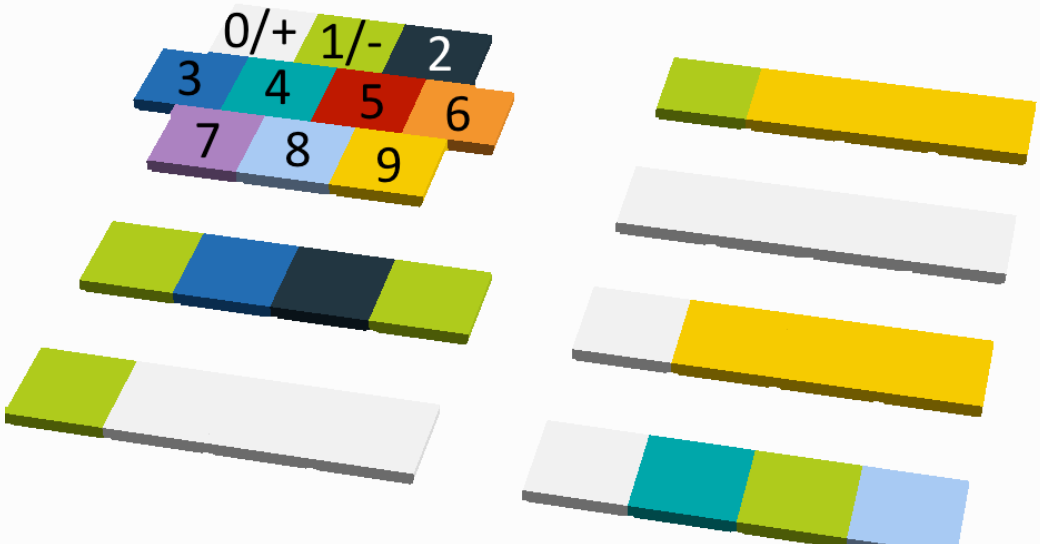
-0,099 à -0,001

0

+0,001 à +0,099

- ▷ Plus de cases ?

Plus de "précisions"



ET DANS UN VRAI ORDINATEUR ?

- ▷ Moins de couleurs ...
  - ▷ Binaire (0 ou 1) – bit
- ▷ C'est quoi le trip avec les Légos alors ?
  - ▷ C'est juste une question d'alphabet ...
  - ▷ Avec 4 bits on peut coder 16 informations

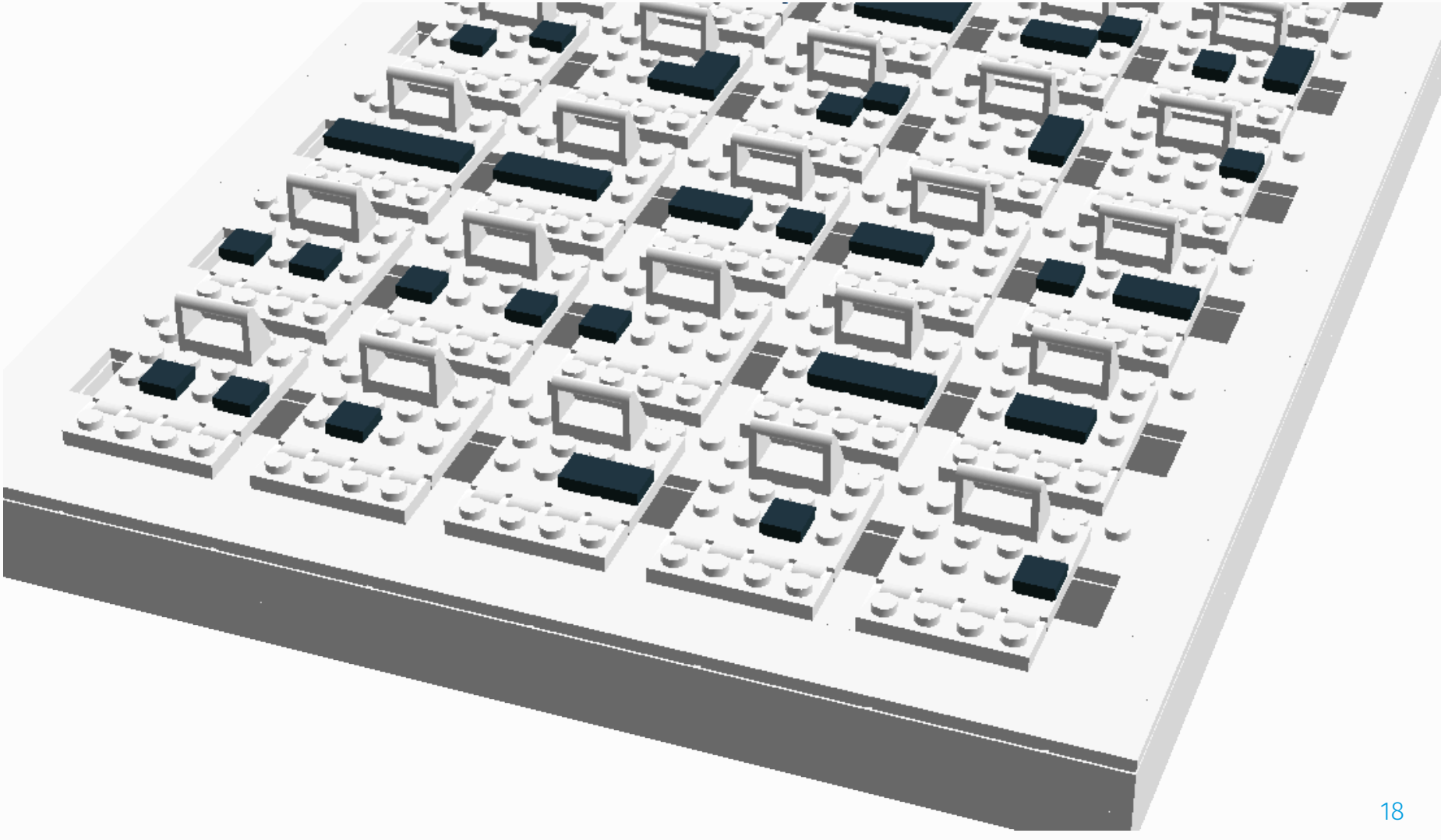
```
0000  0100  1000  1100
0001  0101  1001  1101
0010  0110  1010  1110
0011  0111  1011  1111
```

- ▷ C'est la représentation positionelle en base 2

$$1010_2 = 1*8+0*4+1*2+0*1 = 10_{10} = 1*2^3+0*2^3+1*2^1+0*2^0$$

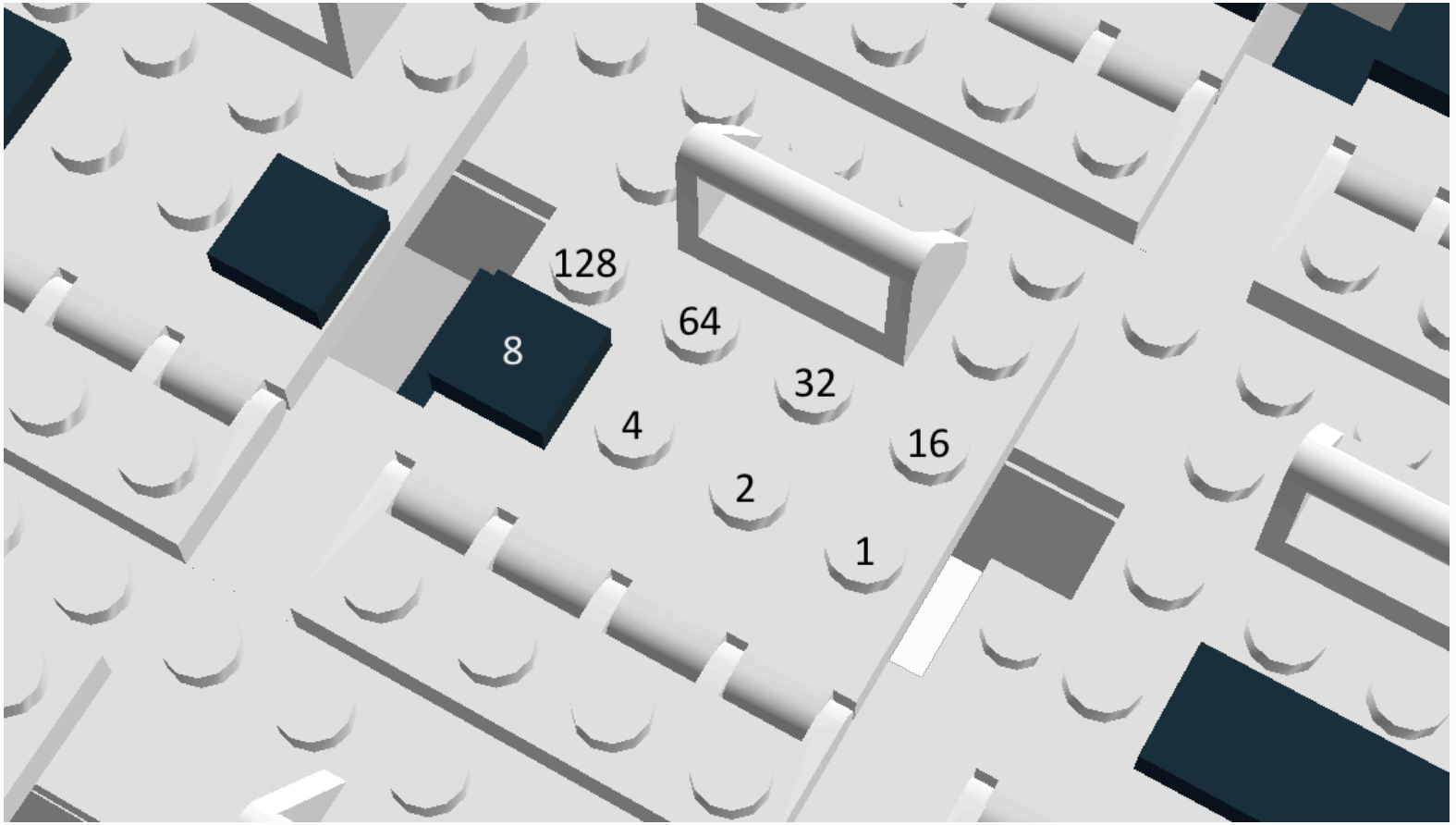
SYSTÈME D'ADRESSAGE

▷ Combinaisons de 8 tuiles (octet ou byte)



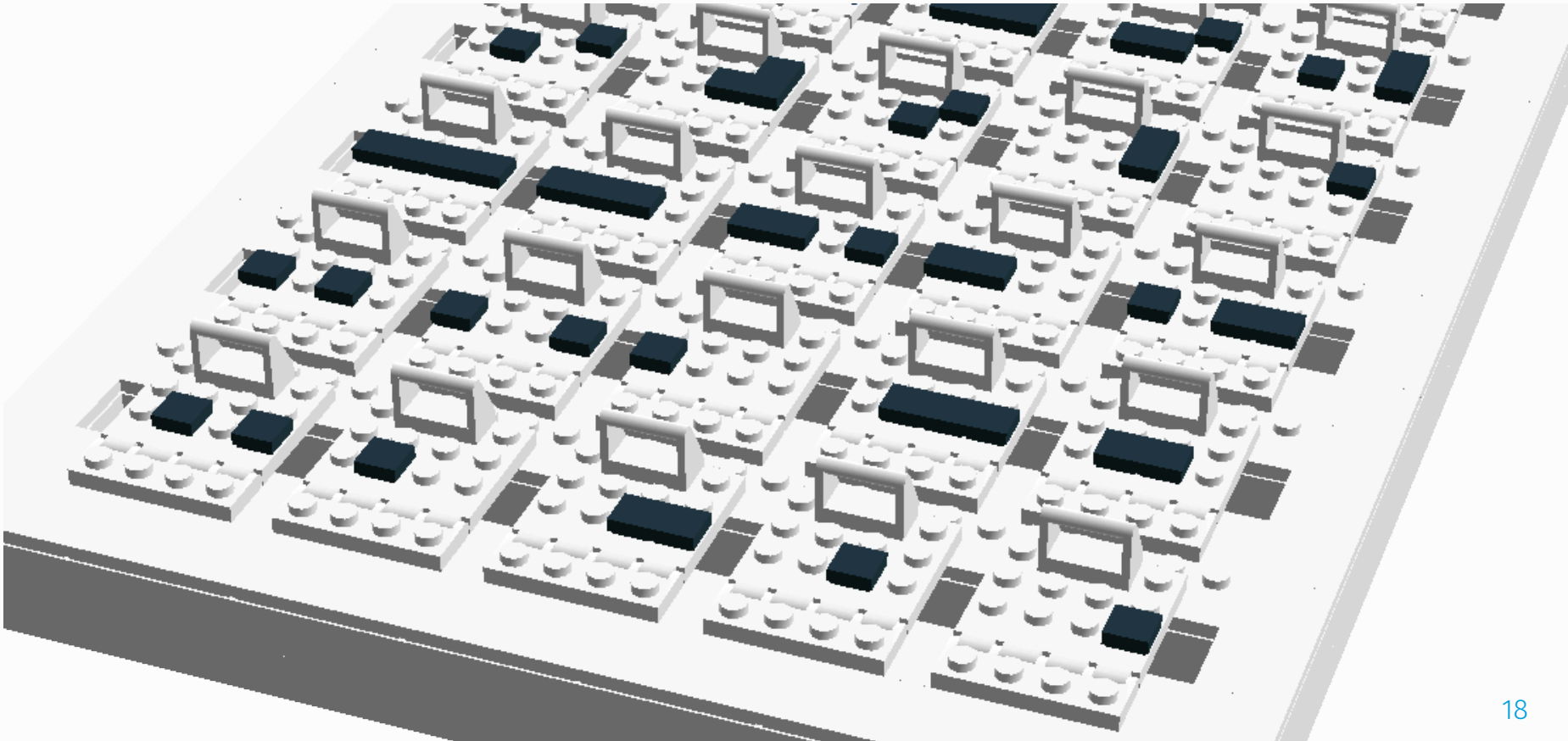
SYSTÈME D'ADRESSAGE

▷ Combinaisons de 8 tuiles (octet ou byte)

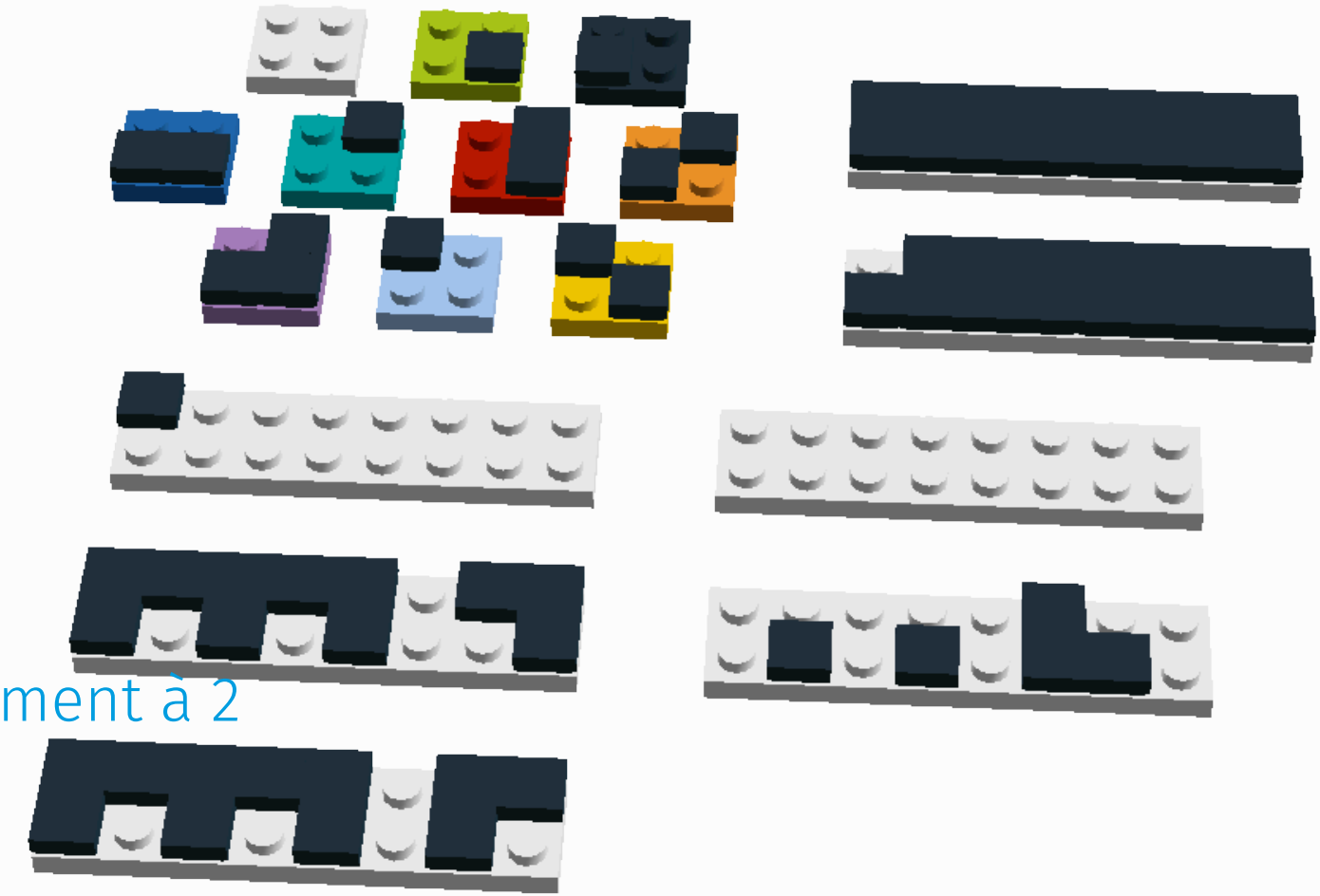


Système d'adressage

- Combinaisons de 8 tuiles (octet ou byte)
  - Consécutivité des adresses (intérêt ?)
    - Une adresse = un entier positif = un indice dans un grand tableau d'octets



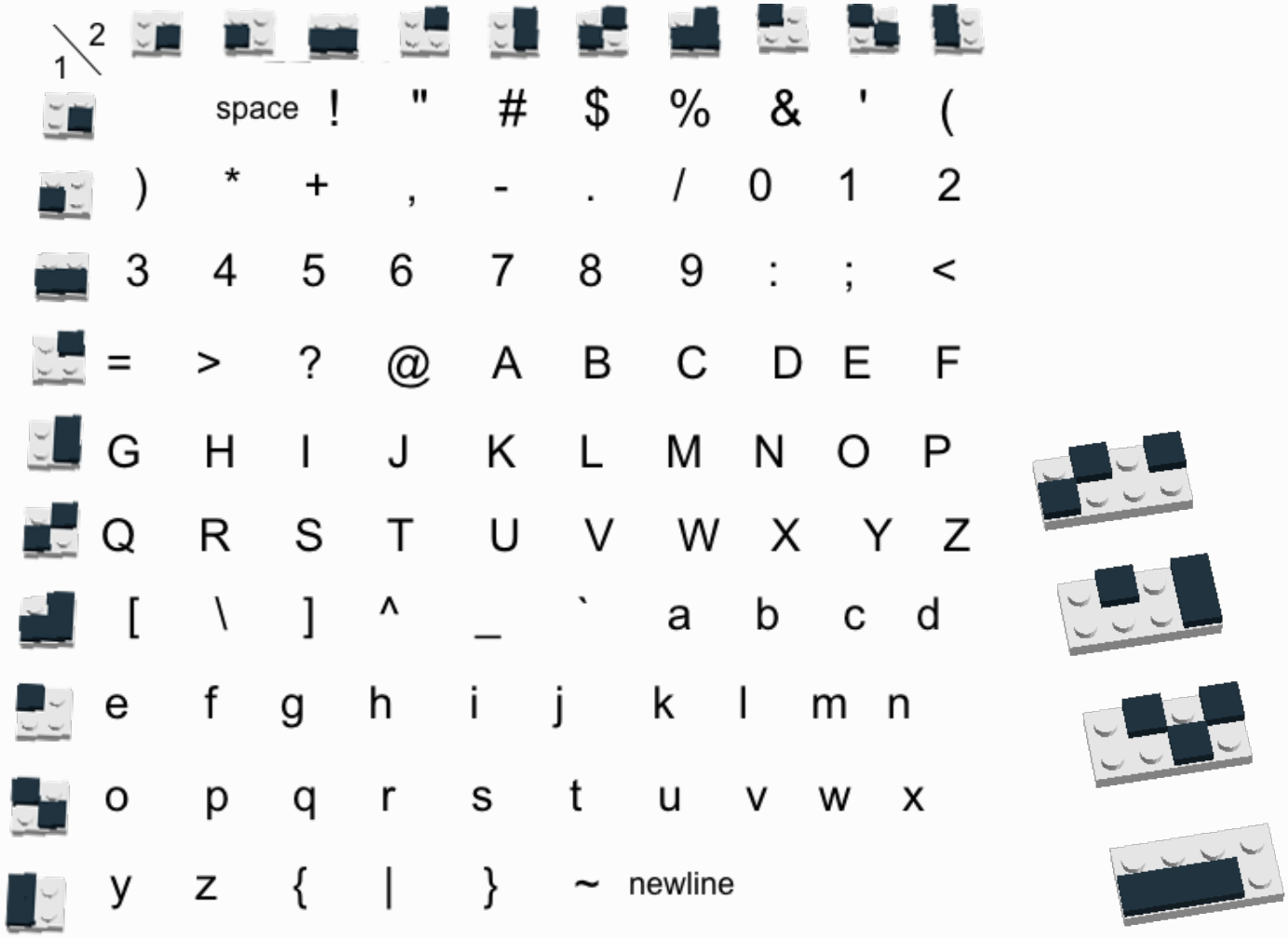
Représentation des entiers sur k bits



- Complément à 2

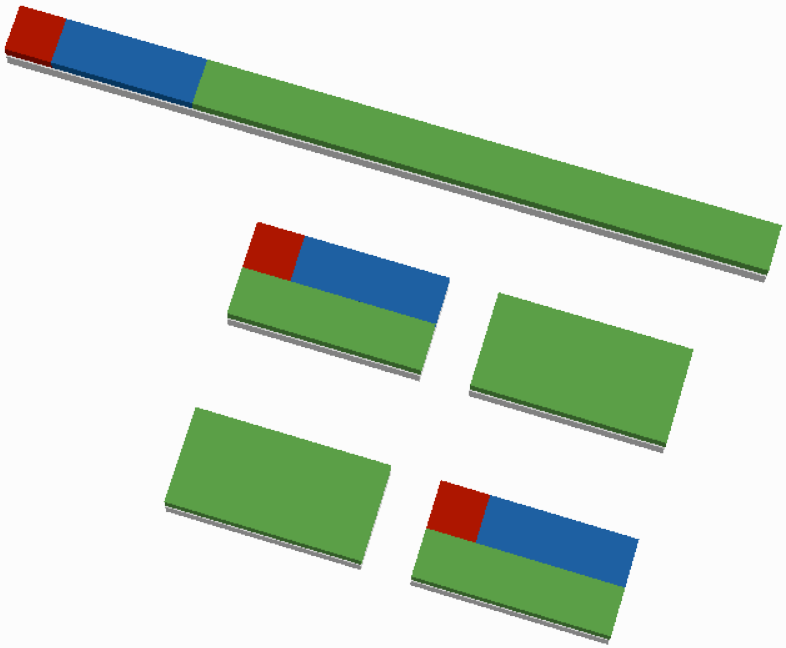
CODAGE D'UN CARACTÈRE

▷ Caractère = entier positif = indice



Représentation à virgule flottante (simplifiée)

- 1 bit pour le signe
- Quelques bits pour la position de la virgule
- Représentation virgule fixe + décalage
- Dans quel ordre les stocker ?
  - Little et Big Endian



RETOUR AU C

LA MÉMOIRE

▷ On manipule toujours de l'espace mémoire à l'aide d'adresse

▷ Chaque adresse correspond à une case mémoire d'un octet (8 bits) = entier positif

▷ La mémoire est compartimentée en zones de stockage

  ▷ Une zone dynamique (pile, tas)

  ▷ Une zone statique (données statiques et le code du programme)

LES TYPES EN C

▷ Type = taille de zone mémoire + interprétation (signé ou non, nombre entier/flottant, caractère ...)

▷ Possiblement dépendant de la machine

▷ Seules contraintes imposées en C : ordre des tailles en mémoire

  ▷ caractere $\leq$ petit entier $\leq$ entier $\leq$ grand entier

LES TYPES EN C

Mot clé	Signification
char	Caractère
unsigned char	Caractère non signé
short int	Entier court
unsigned short int	Entier court non signé
int	Entier
unsigned int	Entier non signé
long int	Entier long
unsigned long int	Entier long non signé
float	Flottant (réel)
double	Flottant double
bool*	Booléen

*C99

TAILLE D'UN TYPE

▷ La fonction **sizeof** permet d'avoir le nombre d'octets occupés par un type donné

```
int main(void){
    printf("Size of short: %d \n", sizeof(short));
    printf("Size of int: %d \n", sizeof(int));
    printf("Size of long: %d \n", sizeof(long));
    printf("Size of float: %d \n", sizeof(float));
    printf("Size of double: %d \n", sizeof(double))
    printf("Size of char: %d \n", sizeof(char));
    return EXIT_SUCCESS ;
}
```

LIMITS.H

Macro	Valeur exemple	Description
CHAR_BIT	8	Number of bits in a byte
SCHAR_MIN	-128	Min value for a signed char
SCHAR_MAX	127	Max value for a signed char
UCHAR_MAX	255	Max value for an unsigned char
CHAR_MIN	0	Min value for type char
CHAR_MAX	127	Max value for type char
SHRT_MIN	-32768	Min value for a short int
SHRT_MAX	+32767	Max value for a short int
USHRT_MAX	65535	Max value for an unsigned short int
INT_MIN	-32768	Min value for an int
INT_MAX	+32767	Max value for an int
UINT_MAX	65535	Max value for an unsigned int
LONG_MIN	-2147483648	Min value for a long int
LONG_MAX	+2147483647	Max value for a long int
ULONG_MAX	4294967295	Max value for an unsigned long int

Représentations des entiers

- En décimal
  - `1234`
- En hexadécimal
  - `0x4D2` ou `0X4d2`
- En octale
  - `02322`
  - en math, `00012=12`, pas en C
- Pas de représentation binaire

Opérateurs sur les entiers

- Addition: `9+4`
- Soustraction: `9-4`
- Multiplication: `9*4`
- Quotient de la division entière: `9/4` (2)
- Reste de la division entière: `9%4` (1)

Problème de débordement

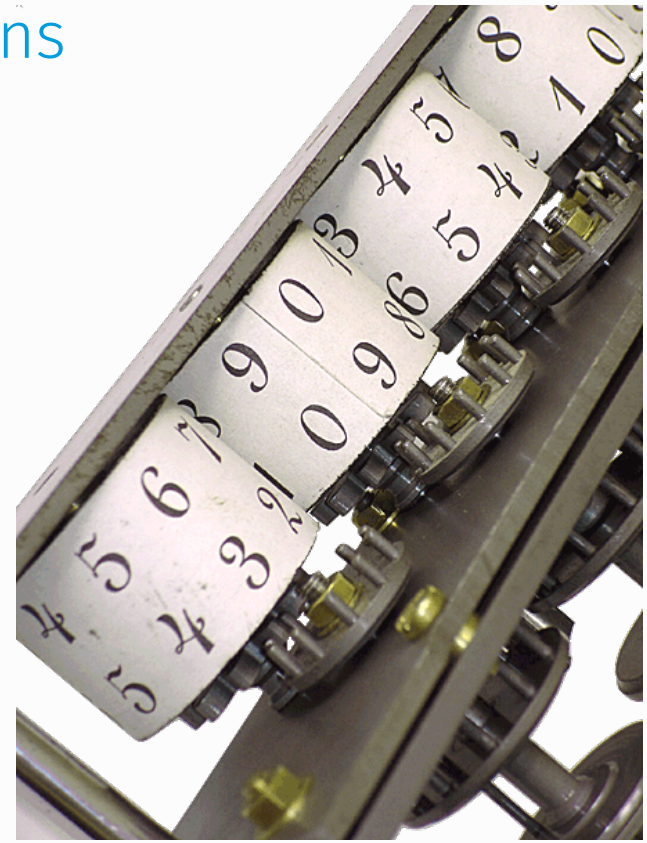
▷ Aucune vérification

▷ Après la valeur représentée par le plus grand code, on trouve la valeur représentée par le plus petit

▷ On peut imaginer les informations représentables sur un rouleau

▷ Peut générer des boucles infinies inattendues

▷ Attendre qu'un `unsigned int` soit strictement inférieur à 0 peut être long ...



LES RÉELS

▷ Calcul en virgule flottante

▷ Approximations (ne pas utiliser pour des calculs exacts, e.g., finances)

▷ `float`: réel simple précision

▷ `double`: réel double précision

Représentations des réels

- ▷ `12.34` ou `1234.` ou `.1234`
- ▷ Notation mantisse $* 10^{\text{exposant}}$
  - ▷ `1723.68 = 1.72368e3 = 17.2368E2`
  - ▷ `0.015 = 1.5e-2`
- ▷ Par défaut, les constantes sont `double`
  - ▷ `245.45f = 245.45F` $\rightarrow$ `float`
  - ▷ `245.45 = 245.45l = 245.45L` $\rightarrow$ `double`
- ▷ Environ 7/15 chiffres significatifs pour les `float/double`

OPÉRATEURS SUR LES RÉELS

- `+ - *` comme pour les entiers
- `/` donne une division réelle
- Si un opérateur a des opérandes de différents types, les valeurs des opérandes sont converties dans un type commun
  - Des types plus "petits" vers les types plus "larges"
  - `2+/-/*3.5` ⇒ `2.0+/-/*3.5`
  - `6/1.5` ⇒ `6.0/1.5`
  - `8.4/2` ⇒ `8.4/2.0`

Problème de précision

▷ Se méfier beaucoup des `float`

  ▷ Une simple somme de `1000` `float` à `0.1f` correspond à `99.999046`

▷ Se méfier quand même des `double`

  ▷ Une simple somme de `10 000 000` `double` à `0.1L` correspond à `999999.999839`

LES CARACTÈRES

- ▷ On interprète les `char` comme des codes de caractères
  - ▷ Pas de problème entre `0` et `127` (commun à tous les encodages)
  - ▷ Au-delà, dépend de l'encodage du système (à éviter)
- ▷ On désigne le code du caractère `x` par `'x'`
- ▷ On peut utiliser les opérateurs entiers sur ces codes
  - ▷ Par exemple, `'a'+1` vaut `'b'`

CARACTÈRES SPÉCIAUX

▷ Attention à ne pas confondre un caractère et la valeur d'un chiffre

  ▷ Le caractère `'9'` n'a pas du tout comme code `9` (mais `57`).

▷ Certains caractères sont spéciaux

  ▷ `'\n'` : saut de ligne

  ▷ `'\r'` : retour au début de ligne

  ▷ `'\t'` : tabulation

  ▷ `'\\'` : le caractère `\`

  ▷ `'\"'` : le caractère `"`

CARACTÈRES ACCENTUÉS

▷ Codes supérieurs à 127

▷ Posent des problèmes de portabilité et d'encodage

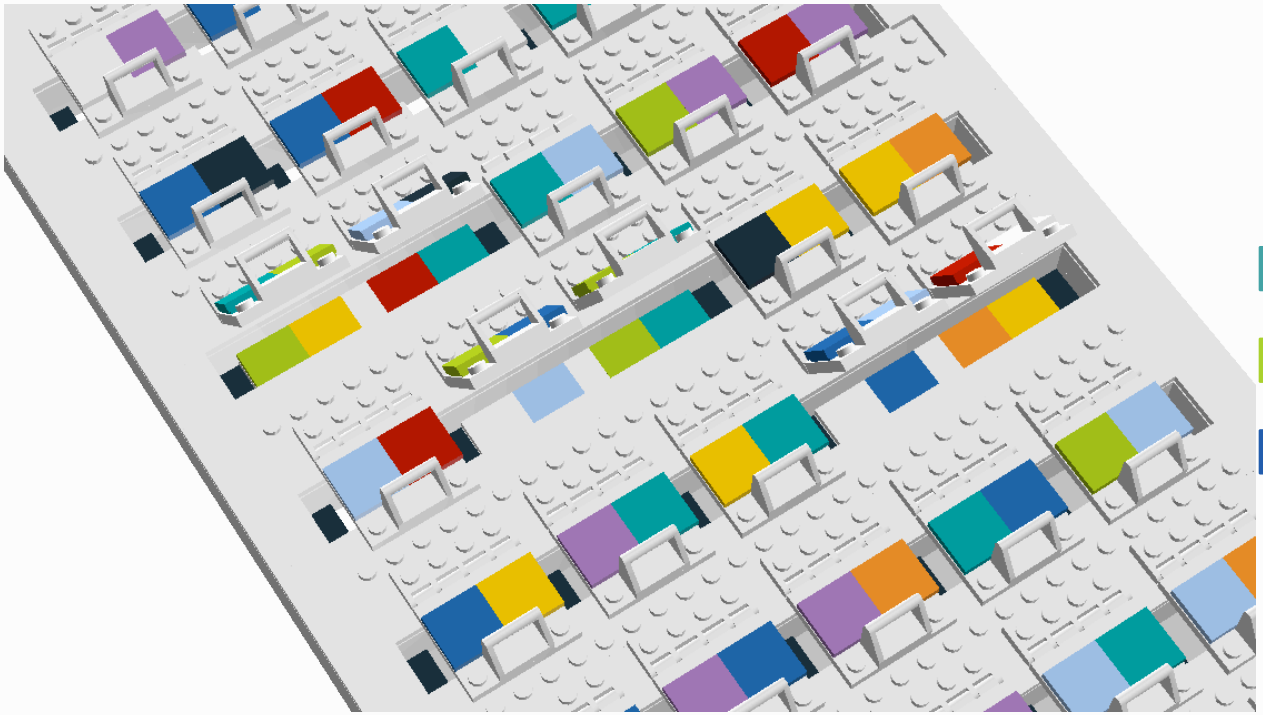
▷ Ne marche plus si on change de système et/ou de codage

▷ Nécessite une gestion fine de la part du programmeur et des éventuelles adaptations des bibliothèques standards de manipulation de chaînes de caractères

LES VARIABLES EN C

▷ Variable = adresse + type + valeur

▷ Pour le programmeur, on lui associe un libellé compréhensible (utile que pour lui car remplacé par l'adresse lors de la compilation)



- int - -954
- char - '
- uint - 369

LES VARIABLES EN C

- Identificateur: `[_a-zA-Z][_a-zA-Z0-9]*`
  - `hello_89` `__tmp` `foo` `bar`
- Eviter ceux ressemblant à des mots-clés (e.g., `new, class`)
- Doivent être déclarées et initialisées[†] avant d'être utilisées
  - Déclaration: `int a=3;`
  - Déclarations multiples: `float f1=0.3,f2=0.0;`

[†] C'est une recommandation

AFFECTATION D'UNE VARIABLE EN C

▷ L'accès à la valeur d'une variable se fait en utilisant son nom directement

▷ La valeur d'une variable peut être modifiée via l'opération d'affectation en utilisant l'opérateur =

```
int a=0, b=0;
a=2+3;
b=a+2;
```

▷ Lors d'une affectation, la donnée à droite du signe d'égalité est convertie dans le type à gauche du signe d'égalité avec possible perte de précision

- ▷ `double a=2;` ⇒ `double a=2.0;`
- ▷ `int a=2.45;` ⇒ arrondi à `int a=2;`

CONVERSION DE TYPES

▷ Possibilité d'interpréter explicitement différemment la valeur d'une variable

▷ Notion de conversion / cast

  ▷ variable = (nouveau_type) ...

```
char a = 'A';
int b = 0;
b = (int) a+1; // b vaut 66
```

▷ ⚠ Il faut être sûr de ce que l'on fait

DOGGY BAG

TO TAKE AWAY ...

- Adresse = entier positif = case mémoire d'un octet
- Type = taille de zone mémoire + interprétation
- Variable = adresse + type + valeur
- Tout est binaire et est une question d'interprétation

Questions?