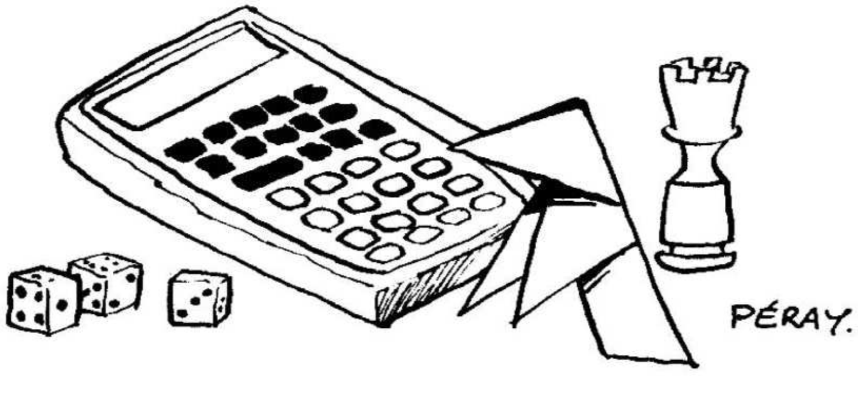
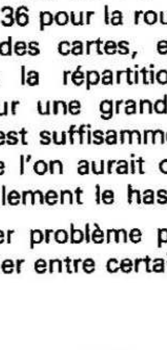


A propos des générateurs de nombres "aléatoires"

L'étude que nous vous proposons de faire ce mois-ci ne nous conduira pas directement à l'élaboration d'un jeu. Il s'agit de mettre au point et de vérifier (succinctement) la qualité d'un élément essentiel à la réalisation d'un jeu : le générateur aléatoire.

■ Les machines électroniques sont, du fait de leur construction même, des « mécaniques déterministes » d'où le hasard a été exclu. Il nous faut donc trouver un moyen simple de « simuler » ce hasard ; notre but initial sera par conséquent d'engendrer des séries de nombres entiers, compris entre certaines limites (de 1 à 6 pour un dé, de 1 à 49 pour le loto, de 0 à 36 pour la roulette, de 1 à 52 pour des cartes, etc.) et de vérifier que la répartition de ces nombres, sur une grande quantité de tirages, est suffisamment proche de celle que l'on aurait obtenue en utilisant réellement le hasard.



Le premier problème posé, fabriquer un entier entre certaines limites est presque toujours résolu d'une manière analogue et très facile à comprendre : en fait, le générateur doit fabriquer un nombre décimal, compris entre 0 et 1, et il suffit de multiplier ce nombre par un entier tel qu'en prenant la partie entière du résultat, on obtienne un entier situé dans les limites fixées. Supposons par exemple que l'on désire obtenir n, $n \in [0,5]$. On part de x, $x \in]0,1[$, et l'on fait : $n = \text{Ent}(6 * x)$, partie entière de 6 x. On obtiendra par exemple :

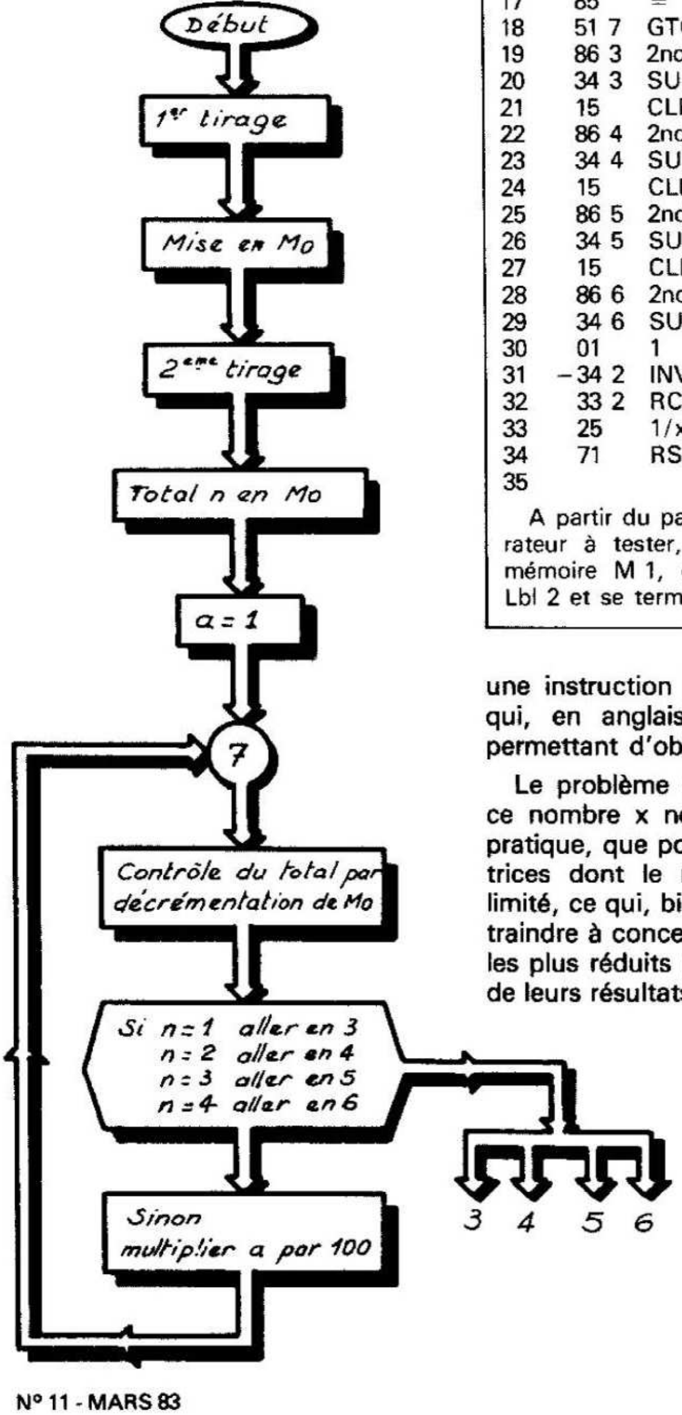
x	0,314	0,825	0,971	0,132
n	1	4	5	0

Si maintenant le résultat doit être situé entre 1 et 6 inclus (cas du jeu de dés), il suffira d'ajouter 1 et chaque nouvelle valeur de x donnera un nombre entre 1 et 6, correspondant au tirage d'un dé. La formule géné-

A propos des générateurs de nombres « aléatoires »

rale permettant d'obtenir un entier n entre a et b, bornes comprises, est donc : n = Ent ((b − a + 1) * x) + a avec x ∈]0,1[. Ainsi, un entier entre 10 et 30 sera obtenu par : n = Ent 21 * x + 10 (avec 21 = 30 − 10 + 1).

Sur un certain nombre de calculatrices évoluées, la connaissance de cette formule est suffisante ; c'est le cas lorsque la machine comporte



Lancer de deux dés avec décompte des totaux obtenus
Programme pour TI 57
Auteur Jacques Deconchat

00	61 2	SBR 2
01	32 0	STO 0
02	61 2	SBR 2
03	34 0	SUM 0
04	01	1
05	86 7	2nd Lbl 7
06	−56	2nd INV Dsz
07	51 3	GTO 3
08	−56	2nd INV Dsz
09	51 4	GTO 4
10	−56	2nd INV Dsz
11	51 5	GTO 5
12	−56	2nd INV Dsz
13	51 6	GTO 6
14	55	*
15	02	2
16	−18	2nd INV Log
17	85	=
18	51 7	GTO 7
19	86 3	2nd Lbl 3
20	34 3	SUM 3
21	15	CLR
22	86 4	2nd Lbl 4
23	34 4	SUM 4
24	15	CLR
25	86 5	2nd Lbl 5
26	34 5	SUM 5
27	15	CLR
28	86 6	2nd Lbl 6
29	34 6	SUM 6
30	01	1
31	−34 2	INV SUM 2
32	33 2	RCL 2
33	25	1/x
34	71	RST
35		

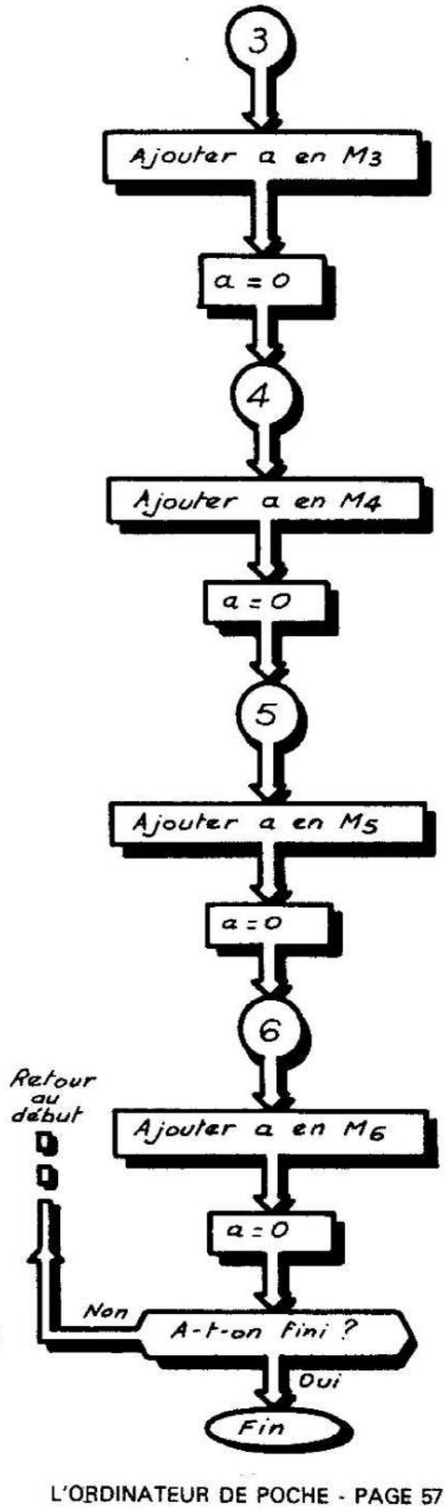
A partir du pas 35, placer le générateur à tester, qui doit utiliser la mémoire M 1, commencer par 2nd Lbl 2 et se terminer par INV SBR.

une instruction (RND pour *random* qui, en anglais, signifie aléatoire) permettant d'obtenir x ∈]0,1[.

Le problème de la confection de ce nombre x ne se pose donc, en pratique, que pour certaines calculatrices dont le nombre de pas est limité, ce qui, bien sûr, va nous contraindre à concevoir des générateurs les plus réduits possibles : la qualité de leurs résultats pourra s'en ressentir. Supposons que nous ayons un générateur délivrant des entiers de 1 à 6 (un dé !) : il faudra d'abord que, sur un grand nombre de tirages, il y ait à peu près autant de 1, que de 2, que de 3... (mais pas exactement autant, ce qui serait « suspect »). Il faut également s'assurer qu'il n'y a aucune périodicité dans la suite des chiffres engendrés (par exemple : 1, 4, 2, 3, 5, 6, 1, 4, 2, 3, 5, 6, 1, 4, 2,...).

La méthode que nous avons retenue pour tester notre générateur consiste à lancer successivement deux dés, un grand nombre de fois, et à stocker en mémoire le nombre

Les flèches numérotées, de 3 à 6 renvoient aux étiquettes correspondantes figurant dans la partie droite de l'organigramme



d'apparitions de chacun des totaux possibles (de 2 à 12). Les résultats théoriques sont consignés dans le tableau ci-dessous.

Total	2	3	4	5	6	7	8	9	10	11	12
Probabilité	1/36	2/36	3/36	4/36	5/36	6/36	5/36	4/36	3/36	2/36	1/36

Si l'on représente graphiquement les résultats obtenus, la courbe a donc de plus en plus de chances de se rapprocher d'un triangle à mesure que l'on augmente le nombre de tirages. Si ce n'est pas le cas, il est probable que votre générateur, ou le noyau choisi au départ pour l'initialiser sont en cause.

Le programme, réalisé sur TI 57, comporte une astuce permettant de stocker les onze résultats différents possibles en utilisant quatre mémoires seulement : on découpera les totaux en tranches de 2 (ou 3 chiffres) de la façon suivante :

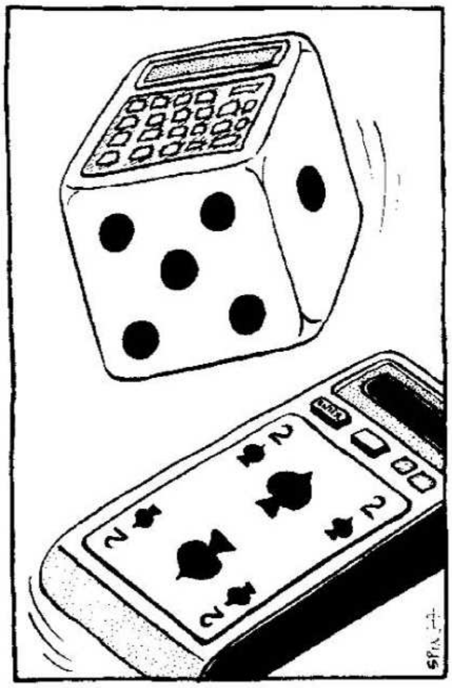
M3	⊔⊔ nbre de 9	⊔⊔ nbre de 5	⊔⊔ nbre de 1
M4	⊔⊔ nbre de 10	⊔⊔ nbre de 6	⊔⊔ nbre de 2
M5	⊔⊔ nbre de 11	⊔⊔ nbre de 7	⊔⊔ nbre de 3
M6	⊔⊔ nbre de 12	⊔⊔ nbre de 8	⊔⊔ nbre de 4

Pour un très grand nombre de tirages (supérieur à 400), il faudra découper en tranches de 3 chiffres (mettre 3 au lieu de 2 au pas 15 du programme), mais le résultat sera plus difficile à lire.

Pour utiliser le programme, on écrira, à partir du pas 35, le générateur qui devra commencer par 2nd Lbl 2 et se terminer par INV SBR, le noyau étant en M1. Entrer le nombre de tirages prévu en mémoire 2 (100 STO 2, par exemple). Faire RST, puis R/S, et attendre le clignotement sur 9.99999...

Comment obtenir simplement un nombre aléatoire de la forme 0,xxxxx... L'idée générale est de fabriquer un nombre à virgule, dont les chiffres suivant la virgule soient répartis de manière imprévisible. De tels nombres existent déjà (Pi, par exemple, mais les machines ne les connaissent qu'avec 10 ou 12 chiffres, et ils ne sont donc pas utilisables tels quels. Une première idée simple, très souvent utilisée par les constructeurs, est de partir d'un noyau quelconque (0,xxxxxxx), de lui ajouter Pi, et d'élever le résultat à une puissance suffisante pour que les chiffres du résultat obtenu puissent être considérés comme imprévisibles : le noyau 0,25167 + Pi donne 3,39326265... qui, élevé à la puissance 8, donne 1757,800248... On prend alors 0,800248 comme nouveau noyau, et l'on peut fabriquer l'entier cherché en prenant x = 0,800248...

La méthode des quotients, utilisée sous différentes formes, est également simple à assimiler : on se sert du fait que les divisions par certains nombres premiers p ont une période de longueur égale à p-1. Ainsi une division par 17 aura, si le nombre initial n'est pas multiple de 17, une période de 16 chiffres. Certains nombres premiers conviennent mieux que d'autres en ce qui concerne la qualité du générateur. Ainsi, si vous divisez un nombre inférieur à 8 000 000 par 8 000 087, vous ne retrouverez le même reste qu'après avoir écrit 8 000 086 chiffres. Une tranche prise au hasard permettra de fabriquer votre nombre, et de recommencer le processus.



Dans de nombreux cas (les jeux de courte durée, par exemple), on pourra simplifier la méthode (au détriment de la qualité du générateur) en prenant des nombres premiers plus petits ou même avec d'autres procédés très simplifiés. Les générateurs suivants, que vous pourrez tester avec le programme précédent, vous donnent quelques exemples :

```
2nd Lbl 2              2nd Lbl 2
RCL 1                  RCL 1
*                      +
9 ┐ ou                 2nd π
9 │ autre              =
7 ┘ (67...)            y^x
=                      8
2nd INV Int            =
STO 1                  2nd INV Int
                       STO 1

2nd Lbl 2              2nd Lbl 2
RCL 1                  2
2nd INV Int            9
*                      2nd Prd 1
2 ┐ ou                 RCL 1
7 ┘ autre              2nd INV Int
=                      STO 1
STO 1
```

D'autres sont encore plus simplifiés (certains ne peuvent même servir qu'une fois) tels que : RCL 1 * π = STO 1, √ 2nd INV Int, ou RCL 1 2nd INV Log 2nd INV Int STO 1.

Bien entendu, pour pouvoir les utiliser dans le programme précédent, il conviendra de ne pas oublier de multiplier par 6, d'ajouter 1 et de prendre la partie entière du résultat. A vos dés et bonne chance.

☐ Jacques Deconchat

Résultats relevés avec le générateur :

```
2nd Lbl 2
RCL 1
2nd INV Log
2nd INV Int
STO 1
*
6
+
1
=
2nd Int
INV SBR
```

Pour 100 essais (100 STO 2), on place le noyau 0,25183 en M1, puis on appuie sur RST et R/S. Au clignotement, on obtient :

RCL 3 : 90600
RCL 4 : 71504
RCL 5 : 42112
RCL 6 : 21307

soit

4 tirages de somme 2
12 tirages de somme 3 ?
7 tirages de somme 4
6 tirages de somme 5 ?
15 tirages de somme 6
21 tirages de somme 7
etc.

Remarque : la mémoire M7 n'est pas utilisée dans le programme proposé. Vous pouvez l'utiliser pour le générateur.