Eskimon

Zeste de savoir

TP Arduino : Faire une animation Space Invaders sur LCD

12 août 2019

Table des matières

Il y a quelques temps, j'avais participé à un petit atelier créatif de programmation pour réaliser une animation de fond d'écran sur le thème « *Space Invaders* ». Le simple but était de s'amuser à coder pour faire bouger un *invader* sur un écran. Comme il n'était pas précisé de quel écran il s'agissait, ni quelles étaient les limites de l'application, je me suis amusé à sortir du cadre « ordinateur = CPU + clavier + écran » et j'ai proposé une solution à base d'Arduino et d'un écran LCD alphanumérique.

Je vous propose dans ce petit tutoriel de voir comment j'ai développé ma solution visible sur la vidéo ci-dessous.

Élément externe (Video) —
Consultez cet élément à l'adresse `https://www.youtube.com/embed/nyFxEGcdvZY?feature=oembed`.

> *i* Si vous débutez sur Arduino, vous pouvez aller relire le chapitre concernant les écrans LCD et Arduino pour vous échauffer.

1. Matériel nécessaire et mise en route

1.1. Le matériel

Pour effectuer ce montage, pas besoin de grand-chose ! Il vous faudra simplement une carte Arduino (Uno dans mon cas) et un écran LCD. Et c'est tout !

1.2. Câbler l'ensemble

Pour ce qui est du câblage, là encore, rien de compliqué. Si vous avez suivi le tutoriel Arduino sur les écrans LCD, il ne devrait pas y avoir de problème. C'est pourquoi je vous propose ici le schéma électronique qui présente comment brancher l'écran en mode « 4 bits » à l'Arduino.

Pour plus d'explications, le tuto est ici .

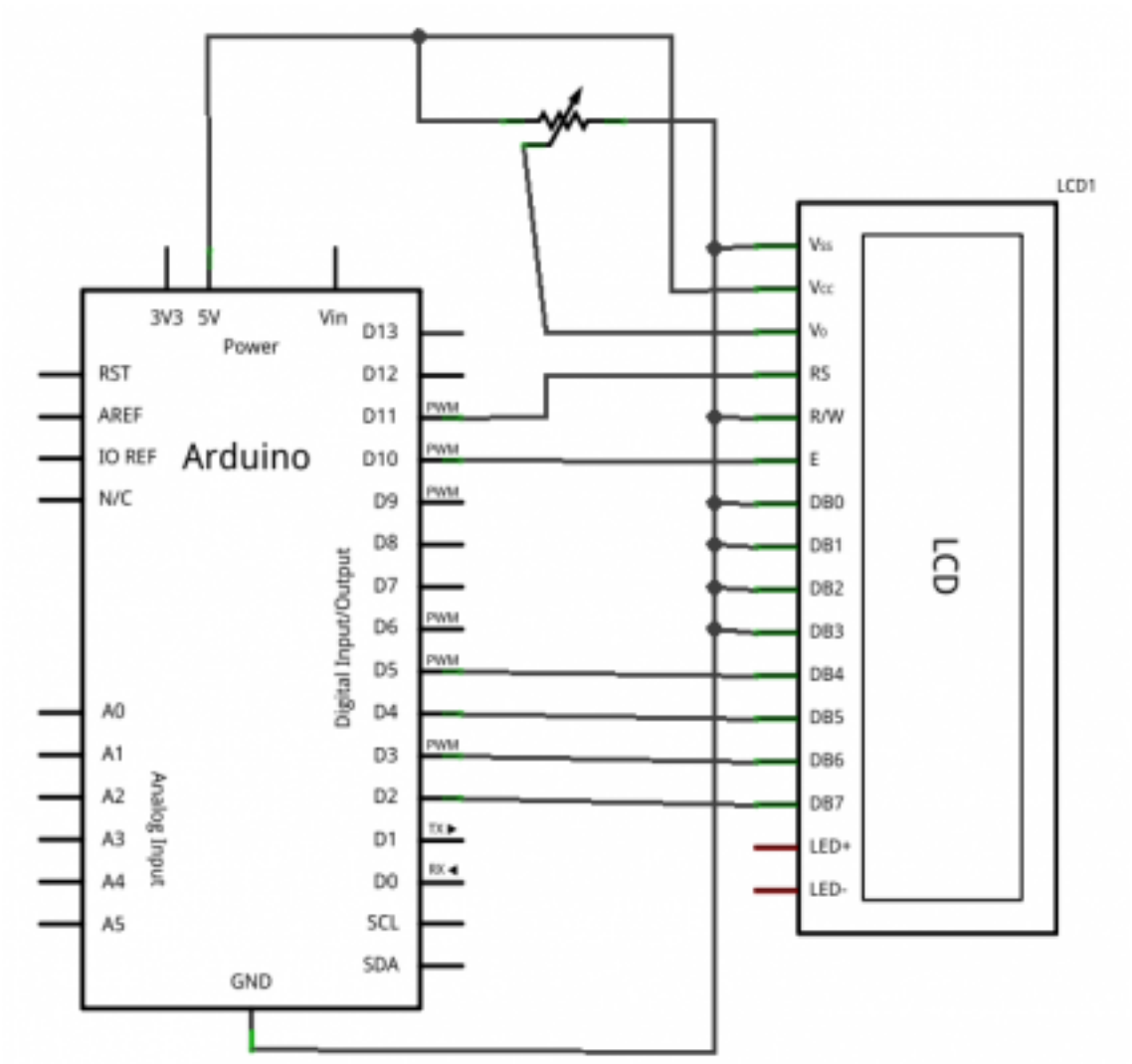


FIGURE 1. – Câblage écran LCD — schéma.

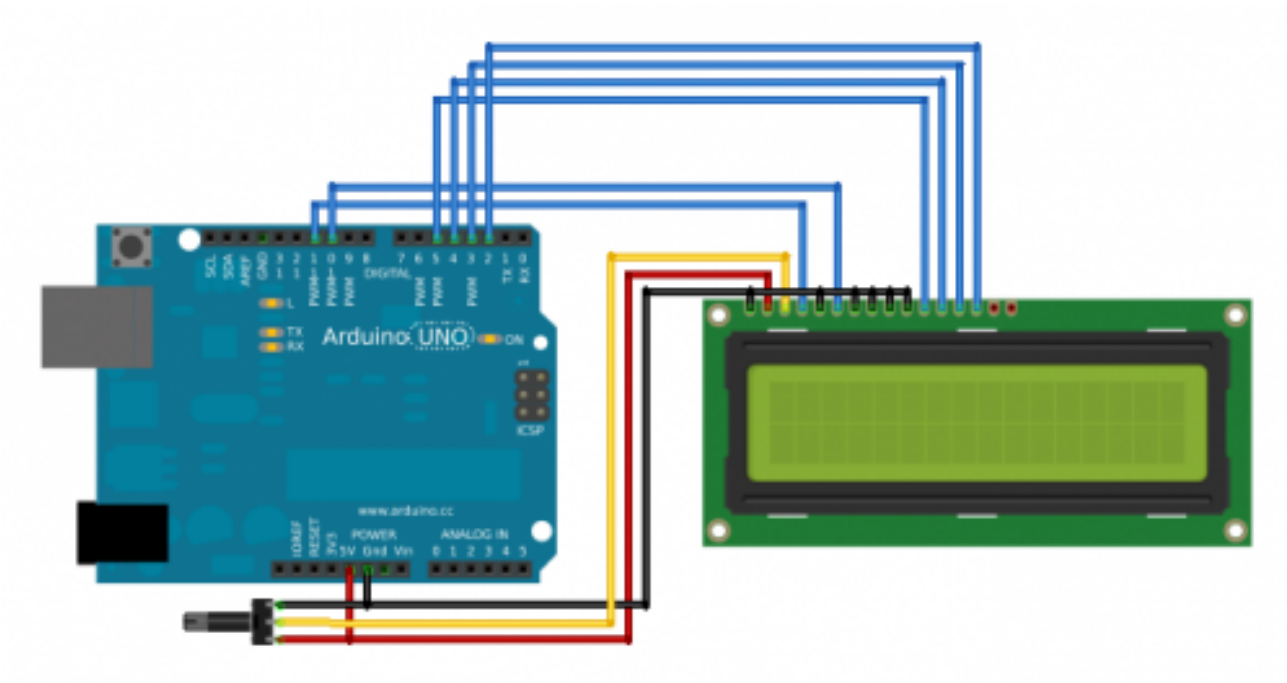


FIGURE 1. – Câblage écran LCD — montage.

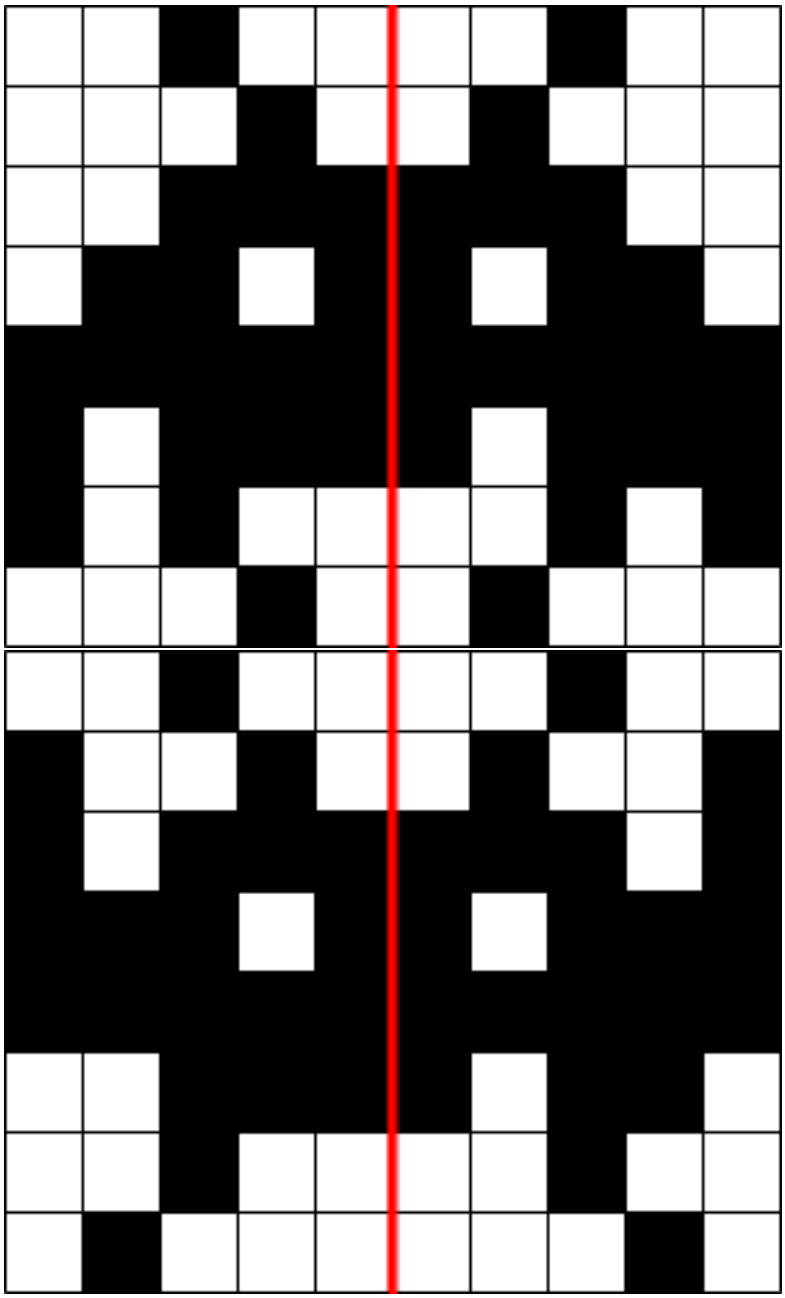
2. Le setup

L'initialisation des variables va avoir une certaine importance ici. En effet, le dessin du petit

```
/opt/zds/data/contents-public/tp-arduino-faire-une-animation-space-in
```

invader n'existe bien entendu pas de base dans l'Arduino ou dans le jeu de caractères du LCD. Il va donc falloir lui « apprendre » à le dessiner. Pour rendre les choses un peu plus sympa, on va le dessiner de deux manières, comme sur le dessin précédent : une fois avec les « bras » baissés et une fois avec les « bras » levés.

Voici un point de vue sous forme de grille de l'envahisseur.



Comme vous pouvez le voir, ces figures ont une taille de 10 pixels de large et 8 de haut, or un caractère affichable par l'écran ne peut avoir qu'une taille de 5 par 8.
Il va donc falloir découper l'**invader** pour l'afficher en deux morceaux côte à côte comme montré avec la ligne rouge sur les images. Ce qui signifie qu'au final, on va créer non pas deux caractères, mais bien quatre. Un pour chacune des parties de chaque *invader*. Si on avait voulu faire une animation avec trois étapes par *invader*, il aurait fallu faire six caractères (et je vous rappelle que l'écran est limité à huit caractères personnalisés).

2.1. Créer les *invaders* dans le code

Si vous ne vous vous souvenez plus trop comment faire des caractères personnalisés, n'hésitez pas à aller relire le tutoriel .

Maintenant que le dessin est prêt, nous allons devoir créer ces caractères dans le code à envoyer à l'Arduino. Ainsi, notre carte pourra « apprendre » à l'écran les nouveaux dessins. Pour cela, on utilise quatre tableaux d'octets (`byte`). Ils sont triés dans l'ordre, de telle façon que chaque paire (0-1 et 2-3) représente les parties gauche et droite de la figure. Chacun de ces tableaux représentera de manière binaire les images vues ci-dessus.

```
// Partie gauche de l'invader bras baissés.
byte invader_low_left[8] = {
  B00100,
  B00010,
  B00111,
  B01101,
  B11111,
  B10111,
  B10100,
  B00010
};
// Partie droite de l'invader bras baissés.
byte invader_low_right[8] = {
  B00100,
  B01000,
  B11100,
  B10110,
  B11111,
  B11101,
  B00101,
  B01000
};
// Partie gauche de l'invader bras levés.
byte invader_high_left[8] = {
  B00100,
  B10010,
  B10111,
  B11101,
  B11111,
  B01111,
  B00100,
  B01000
};
// Partie droite de l'invader bras levés.
byte invader_high_right[8] = {
  B00100,
```

```
  B01001,
  B11101,
  B10111,
  B11111,
  B11110,
  B00100,
  B00010
};
```

2.2. Le reste du *setup*

Voilà, le plus dur est fait, il ne nous reste plus qu'à intégrer les bricoles qui font que notre LCD va comprendre ce qu'on lui dit.

Pour cela, on va, dans l'ordre :

1. Ajouter la librairie `<LiquidCrystal.h>` en haut de notre fichier ;
2. Déclarer un objet `LiquidCrystal` avec les bonnes broches à utiliser ;
3. Dans le `setup()`, envoyer les quatre caractères au LCD (en mode « commande ») ;
4. Démarrer la communication avec le LCD (passage en mode « écriture ») ;
5. Démarrer le générateur de nombres aléatoires pour faire bouger l'*invader*.

Cela pourra se traduire en code Arduino de la manière suivante.

```
// 1. On inclut la librairie LiquidCrystal.
#include <LiquidCrystal.h>

// 2. On crée un objet LiquidCrystal (nommé « lcd » dans notre
    cas).
LiquidCrystal lcd(8, 9, 5, 4, 3, 2);

void setup() {
  /*
     Ne pas oublier de créer les quatre tableaux
     pour les caractère représentant les invaders
     (omis ici pour gagner de la place/visibilité).
  */

  // 3. On envoie les nouveaux caractères à l'écran.
  lcd.createChar(0, invader_low_left);   // Apprend le caractère à
      l'écran LCD.
  lcd.createChar(1, invader_low_right);  // Apprend le caractère à
      l'écran LCD.
  lcd.createChar(2, invader_high_left);  // Apprend le caractère à
      l'écran LCD.
```

```
  lcd.createChar(3, invader_high_right); // Apprend le caractère à
      l'écran LCD.

  // 4. On passe l'écran en mode « écriture ».
  lcd.begin(16, 2);

  // 5. On démarre le générateur de nombres aléatoires.
  randomSeed(analogRead(0)); // Initialise l'aléatoire avec une
      lecture analogique.

  // 6. Bonus, on dit coucou à l'utilisateur, puis on efface.
  lcd.write("Hello World!");
  delay(2000);
  lcd.clear();
}
```

Le code complet du *setup* avec la création des caractères.

Contenu masqué n°1

3. Animation et code complet

Maintenant que tout est mis en place, il ne reste plus qu'à réaliser l'animation proprement dite.

Pour faire les choses proprement, on va écrire une fonction `drawInvader()` pour dessiner. Cette fonction prendra en paramètre la ligne et la colonne où l'*invader* doit être dessiné, et bien sûr l'état de l'*invader* (bras levés ou baissés).

Comme vous avez pu le voir dans le *setup*, on va utiliser une fonction permettant de générer des nombres aléatoires pour déterminer la position du vaisseau. Cette fonction se nomme tout simplement `random()` et prend en paramètre la borne inférieure (incluse) et la borne supérieure (non incluse) du nombre aléatoire à retourner (un `long`). Dans notre cas, en vertical, on veut pouvoir obtenir 0 ou 1, et en horizontal, 0 à 14 (car si on fait 15, la seconde moitié de l'*invader* ne sera pas affichée). On va donc faire ce qui suit.

```
char ligne = random(0, 2);
char colonne = random(0, 15);
```

La suite de la *loop* est assez simple. Une fois que l'on a les coordonnées, on appelle la fonction `drawInvader` une première fois avec l'état « bras baissés », on fait une petite pause, puis on refait l'appel avec les bras levés pour obtenir l'animation complète. De nouveau une petite pause, puis le cycle repart du début.

Voici le code complet de la boucle.

```
void loop() {
  char ligne = random(0, 2);
  char colonne = random(0, 15);
  drawInvader(ligne, colonne, 0); // État 0 : bras baissés.
  delay(800);
  drawInvader(ligne, colonne, 1); // État 1 : bras levés.
  delay(800);
  lcd.clear();
}
```

? C'est bien beau mais `drawInvader` dans tout ça ?

Une difficulté ? Où ça ? Si vous savez comment afficher un caractère personnalisé sur l'écran, alors il ne devrait y avoir aucun problème.

En effet, la suite logique des opérations sera la suivante :

1. On efface l'écran ;
2. On place le curseur au bon endroit ;
3. On affiche la moitié gauche ;
4. On affiche la moitié droite (logiquement à la suite).

Et en code, voici ce que cela donnerait.

```
void drawInvader(char ligne, char colonne, char etat)
{
  lcd.clear(); // Efface l'écran.
  lcd.setCursor(colonne, ligne);   // Se place au bon pixel.
  lcd.write((uint8_t) (0+etat*2)); // Affiche la moitié gauche.
  lcd.write((uint8_t) (1+etat*2)); // Affiche la moitié droite.
}
```

`etat` sera la variable qui dira si les bras sont levés ou baissés. Comme les caractères sont enregistrés dans l'ordre (bras baissés gauche, bras baissés droite, bras levés gauche, bras levés droite), il suffit de faire une multiplication par 2 pour se situer au bon endroit. On ajoute 0 ou 1 si on envoie la moitié gauche ou droite (gauche : 0, droite : 1).

Et voila ! Vous avez maintenant tout le code pour faire une jolie animation *Space Invaders* sur votre écran LCD et donner un petit côté années 80 à votre Arduino !

Contenu masqué n°2

Une démonstration live est visible sur le lient suivant : https ://www.tinkercad.com/embed/5iymNgseWcN

!(https ://www.tinkercad.com/embed/5iymNgseWcN)

Bien sûr, comme pour tout code, des améliorations pourraient être faites. Par exemple, on pourrait utiliser des constantes avec des `#define` pour éviter d'envoyer 0 ou 1 pour l'état. On écrirait ainsi `BAISSE` ou `LEVE` et ça serait déjà plus propre.

Si vous le souhaitez, en guise d'exercice d'approfondissement, vous pouvez essayer de rajouter une étape intermédiaire où les bras du vaisseau seront à l'horizontale. Attention, il faudra peut-être revoir l'ordre d'enregistrement des caractères dans la mémoire de l'écran, donc le calcul pour les appeler. Amusez-vous bien et n'hésitez pas à poser des questions ou faire des remarques en commentaires.

i

Un gros merci à Dominus Carnufex qui, comme souvent sur mes contenus, à eu l'immense loisir de s'écorcher la rétine sur mes fautes .

Contenu masqué

Contenu masqué n°1

```
1  // 1. On inclut la librairie LiquidCrystal.
2  #include <LiquidCrystal.h>
3
4  // 2. On crée un objet LiquidCrystal (nommé « lcd » dans notre
       cas).
5  LiquidCrystal lcd(8, 9, 5, 4, 3, 2);
6
7  void setup() {
8
9    // Partie gauche de l'invader bras baissés.
10   byte invader_low_left[8] = {
11     B00100,
12     B00010,
13     B00111,
14     B01101,
15     B11111,
16     B10111,
17     B10100,
18     B00010
19   };
20   // Partie droite de l'invader bras baissés.
21   byte invader_low_right[8] = {
```

Contenu masqué

```
    B00100,
    B01000,
    B11100,
    B10110,
    B11111,
    B11101,
    B00101,
    B01000
  };
  // Partie gauche de l'invader bras levés.
  byte invader_high_left[8] = {
    B00100,
    B10010,
    B10111,
    B11101,
    B11111,
    B01111,
    B00100,
    B01000
  };
  // Partie droite de l'invader bras levés.
  byte invader_high_right[8] = {
    B00100,
    B01001,
    B11101,
    B10111,
    B11111,
    B11110,
    B00100,
    B00010
  };

  // 3. On envoie les nouveaux caractères à l'écran.
  lcd.createChar(0, invader_low_left);   // Apprend le caractère à
      l'écran LCD.
  lcd.createChar(1, invader_low_right);  // Apprend le caractère à
      l'écran LCD.
  lcd.createChar(2, invader_high_left);  // Apprend le caractère à
      l'écran LCD.
  lcd.createChar(3, invader_high_right); // Apprend le caractère à
      l'écran LCD.

  // 4. On passe l'écran en mode « écriture ».
  lcd.begin(16, 2);

  // 5. On démarre le générateur de nombres aléatoires.
  randomSeed(analogRead(0)); // Initialise l'aléatoire avec une
      lecture analogique.

  // 6. Bonus, on dit coucou à l'utilisateur, puis on efface.
```

```
  lcd.write("Hello World!");
  delay(2000);
  lcd.clear();
}
```

Retourner au texte.

Contenu masqué n°2

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(8, 9, 5, 4, 3, 2);

void setup() {

  // Partie gauche de l'invader bras baissés.
  byte invader_low_left[8] = {
    B00100,
    B00010,
    B00111,
    B01101,
    B11111,
    B10111,
    B10100,
    B00010
  };
  // Partie droite de l'invader bras baissés.
  byte invader_low_right[8] = {
    B00100,
    B01000,
    B11100,
    B10110,
    B11111,
    B11101,
    B00101,
    B01000
  };
  // Partie gauche de l'invader bras levés.
  byte invader_high_left[8] = {
    B00100,
    B10010,
    B10111,
    B11101,
    B11111,
    B01111,
    B00100,
```

Contenu masqué

```
    B01000
  };
  // Partie droite de l'invader bras levés.
  byte invader_high_right[8] = {
    B00100,
    B01001,
    B11101,
    B10111,
    B11111,
    B11110,
    B00100,
    B00010
  };

  lcd.createChar(0, invader_low_left);   // Apprend le caractère à
      l'écran LCD.
  lcd.createChar(1, invader_low_right);  // Apprend le caractère à
      l'écran LCD.
  lcd.createChar(2, invader_high_left);  // Apprend le caractère à
      l'écran LCD.
  lcd.createChar(3, invader_high_right); // Apprend le caractère à
      l'écran LCD.

  lcd.begin(16, 2);

  randomSeed(analogRead(0)); // Initialise l'aléatoire avec une
      lecture analogique.

  lcd.write("Hello World!");
  delay(2000);
  lcd.clear();
}

void loop() {
  char ligne = random(0, 2);
  char colonne = random(0, 15);
  drawInvader(ligne, colonne, 0);
  delay(800);
  drawInvader(ligne, colonne, 1);
  delay(800);
  lcd.clear();
}

void drawInvader(char ligne, char colonne, char etat)
{
  lcd.clear(); // Efface l'écran.
  lcd.setCursor(colonne, ligne);   // Se place au bon pixel.
  lcd.write((uint8_t) (0+etat*2)); // Affiche la moitié gauche.
  lcd.write((uint8_t) (1+etat*2)); // Affiche la moitié droite.
}
```

Contenu masqué

Retourner au texte.