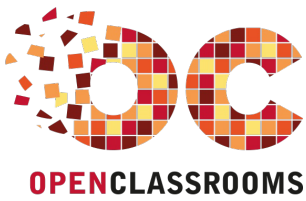
[TP] Pendu

Par Triviak



www.openclassrooms.com

Dernière mise à jour le 14/06/2010

Sommaire

[TP] Pendu

Par  Triviak

Mise à jour : 14/06/2010
Difficulté : Facile

Bienvenue ! Le but de ce tutoriel est de créer un Pendu comme dans le tuto de M@teo21 sauf qu'il sera réalisé en SDL.

Avant de commencer à lire ce tutoriel, vous devez avoir lu celui de M@teo21 jusqu'au chapitre 8 soit le chapitre sur SDL_ttf.

L'intérêt de ce tutoriel est de créer un programme en SDL avec l'utilisation de la bibliothèque SDL_ttf. Si vous venez de lire le tuto sur SDL_ttf, alors le jeu du pendu est un très bon exercice. Et même si vous savez utiliser SDL_ttf, venez tout de même faire un pendu !!

N'attendons pas plus longtemps, c'est parti !
Sommaire du tutoriel :

Objectifs

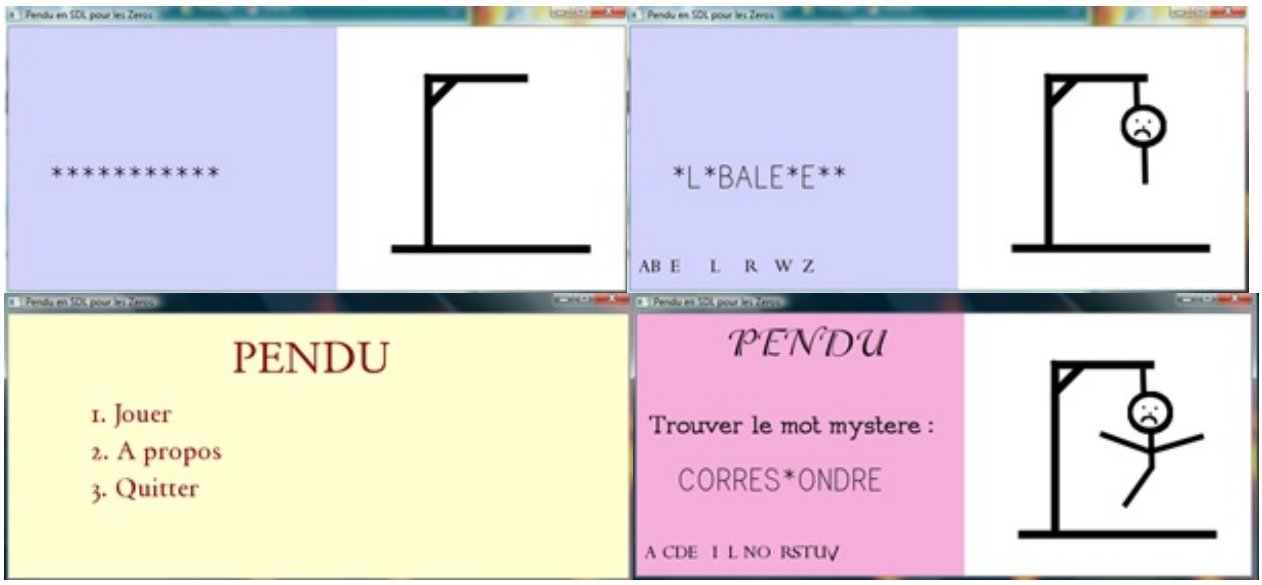
REGLES DU JEU

Pour ceux qui n'auraient pas compris, on veut créer un pendu en SDL.
Hop, un petit rappel des règles pour que l'on fasse le même jeu. Le but est de trouver un mot qui est masqué. Le joueur va donc essayer de découvrir le mot en tapant une lettre. Cependant, le joueur n'a que sept essais. En effet, si le joueur tape un A et que le mot secret en contient effectivement, le joueur ne perd pas d'essais et on découvre tous les A du mot secret. Cependant, si le mot ne contenait pas de A, le joueur perd un essai. Voilà un petit rappel du jeu.

Mais pourquoi on appelle-t-on ce jeu le pendu ?

Car à chaque fois que le joueur perd un essai, un dessin se complète en formant **un bonhomme en train de se faire pendre** ! sisi !! En effet, lorsque vous avez réalisé le pendu en console, je ne crois pas que vous aviez envie de créer un petit bonhomme rien qu'avec des | , / , et \ surtout quand vous êtes justement arrivé à la fin de la partie II.

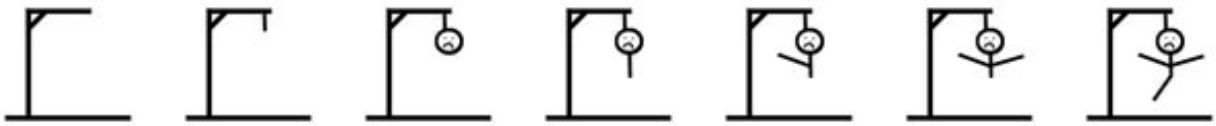
Je vous donne tout de suite quatre écrans obtenus. L'objectif est de réaliser un pendu qui ressemble plus ou moins à cela.



Comme vous pouvez le voir, il y a deux versions de pendu. Et oui, comme tout tutoriel qui se respecte, je vais vous donner à la fin de celui-ci **quelques améliorations** que vous pourrez faire. Les deux images du dessus représentent le pendu que nous allons réaliser. Les deux autres images représentent le pendu avec les améliorations.

REALISATIONS

Voici les images dont j'ai eu besoin. Je tiens à signaler que je n'ai pas un très bon niveau avec photoshop.



Ne dites rien sinon je vais me vexer  et gare à vous  .

J'ai créé un fichier zip avec toutes les images, cela vous fera gagner un peu de temps. Mais rien ne vous empêche de créer vos propres images... J'ai aussi rajouté un dictionnaire de plus de 22 000 mots en majuscule avec un retour à la ligne entre chaque mot. Enfin, j'ai ajouté trois polices.
De plus, je me suis permis de rajouter le mot **"ZERO"** qui me semble tout à fait approprié dans ce dictionnaire.

Télécharger le pack d'image.

Comme vous le voyez sur les images, mon pendu est constitué de deux parties. À gauche le mot secret et en dessous de lui les lettres qui ont déjà été essayées, puis à droite on a les images. Vous pouvez voir sur la capture d'écran que les lettres en dessous du mot secret sont classées dans l'ordre alphabétique. Vous pouvez très bien modifier cela, mais la correction sera adaptée à cette copie d'écran.

Maintenant, vous avez deux choix :

- Soit vous commencez à coder le jeu tout de suite. Alors ne lisez pas plus loin.
- Soit vous vous aidez de ce que vous avez fait lorsque vous avez réalisé ce programme en console. Alors vous pouvez continuer. Je vous donnerais aussi quelques conseils concernant "le plan" du code à réaliser.

Je vais donc vous redonner ici les fonctions qui nous ont servi lors du précédent ~~épisode~~ pendu.

Code : C

```
srand(time(NULL));
nombreMystere = (rand() % (MAX - MIN + 1)) + MIN;
```

On fait deux trois petits changements et on obtient une fonction qui renvoi un int au hasard entre min et max.

Code : C

```
int nombreAleatoire(int max, int min);
int main(int argc, char *argv[])
{
 srand(time(NULL));
 intAuHasard = nombreAleatoire(10, 1500);
}
int nombreAleatoire(int max, int min)
{
    return (rand() % (max - min + 1)) + min;
}
```

N'y aurait-il pas une autre fonction qui pourrait nous aider ?

Oui. C'est bien beau d'avoir un dictionnaire mais encore faut-il trouver un mot dedans, qui plus est au hasard. 
Voici donc pour vous :

Code : C

```
int piocherMot(char *motPioche);

int main(int argc, char *argv[])
{
...
}

int piocherMot(char *motPioche)
{
    FILE* dico = NULL;
    int nombreMots = 0, numMotChoisi = 0;
    int caractereLu = 0;
    dico = fopen("sources/dico.txt", "r"); // ATTENTION à
l'emplacement
    if (dico == NULL)    //
    {
        return 0;
    }
    do
    {
        caractereLu = fgetc(dico);
        if (caractereLu == '\n')
            nombreMots++;
    }
    while (caractereLu != EOF);
    numMotChoisi = nombreAleatoire(nombreMots);
    rewind(dico);
    while (numMotChoisi > 0)
    {
        caractereLu = fgetc(dico);
        if (caractereLu == '\n')
            numMotChoisi--;
    }
    fgets(motPioche, 100, dico);
```

```
        motPioche[strlen(motPioche) - 1] = '\0';
        fclose(dico);
        return 1;
    }
```

La fonction piocherMot renvoi 0 si elle n'arrive pas à charger le dictionnaire et renvoi 1 si tout se passe bien. Elle prend en paramètre un tableau de chars.

Je tiens à signaler que l'objectif n'est pas de créer un pendu mais bien de créer **un pendu EN SDL**. Je suppose donc que vous avez déjà compris ces deux fonctions lors des deux premières parties du tutoriel de M@teo21. Si ce n'est pas le cas, on retourne voir son cours : nombre aléatoire et lire et écrire dans un fichier.

Plan du code

Vous ne savez pas comment démarrer ? Vous n'arrivez pas à visualiser le code avant de vous lancer sur votre IDE ?
Pas de problème. Voici comment j'ai fait pour réaliser ce code.

- Première étape : le squelette. Il faut créer le code de base : afficher une fenêtre, insérer les deux fonctions ci-dessus, prévoir la place du switch qui gère les évènements...
- Deuxième étape : Création des variables. Cette étape permet de visualiser votre futur code. Si vous connaissez le nombre et le type de variables que vous avez besoin, alors vous pouvez vous lancer. Cette étape est vraiment importante car elle permet de représenter plus ou moins le fonctionnement de votre programme.
- troisième étape : la boucle des évènements.
- quatrième étape : Le contenu.
    - Dans un premier temps, il faut s'assurer que le joueur a bien tapé une lettre, et aussi qu'il ne l'avait pas déjà essayé. Il vous faudra donc explorer un tableau de 26 caractères pour s'en assurer. Vous pourrez par exemple réaliser ceci à l'aide de 2 conditions de type if.
    - A partir de maintenant, le joueur tente une lettre. Il vous faut donc gérer le nombre de ses essais. Petit rappel, si le joueur a trouvé au moins une lettre, on ne lui retire pas son essais.
    - Le nombre d'essais étant fixé, on peut donc définir l'image à blitter, et aussi savoir si le joueur a gagné, a perdu, ou peut encore continuer à jouer.
    - On blitte tout ! Le texte (le mot secret, les lettres déjà essayées, ...), mais aussi les images. Vous devrez blitter différentes choses si le joueur a gagné ou a perdu.
- Et on n'oublie pas de libérer la mémoire.

Et voila, maintenant c'est à vous de travailler.
Bon courage les Zéros !!! 

Correction

Fini ou pas fini, si vous lisez ces lignes, c'est parce que vous voulez la correction. Et justement elle arrive ! 
Pour des raisons pratiques j'ai préféré n'utiliser que le fichier main.c. Mais rien ne vous empêche de mettre les deux fonctions dans un .h et un .c et de recréer d'autres fonctions.
Voila comment on va procéder. Je vais découper mon code en plein de petites parties que je vais expliquer séparément. Etant donné que j'ai bien décrit l'objectif final dans la première partie, vous devriez comprendre.
Et maintenant, c'est parti ! 

Créer la fenêtre

Tout d'abord, on va créer une fenêtre toute simple avec les deux fonctions que j'ai donné dans la partie "objectifs". On inclut d'avance SDL_ttf car on en aura besoin. N'oubliez pas de linker votre bibliothèque pour pouvoir utiliser SDL_ttf.

Code : C

```
#include <stdlib.h>
#include <stdio.h>
```

```
#include <string.h>
#include <SDL/SDL.h>
#include <time.h>
#include <SDL_image.h>
#include <SDL/SDL_ttf.h>

int piocherMot(char *motPioche);
int nombreAleatoire(int nombreMax);

int main(int argc, char *argv[])
{
    SDL_Surface *ecran = NULL;
    SDL_Event event;
    srand(time(NULL));
    int continuer = 1;

    SDL_Init(SDL_INIT_VIDEO);
    ecran  = SDL_SetVideoMode(750, 320, 32, SDL_HWSURFACE |
SDL_DOUBLEBUF);

    TTF_Init();

    SDL_WM_SetCaption("Pendu en SDL pour les Zeros", NULL);

    while (continuer)  //Boucle des évenements
    {
        SDL_WaitEvent(&event);
        switch (event.type)
        {
        case SDL_QUIT:
            continuer = 0;
            break;
        case SDL_KEYDOWN:
            switch (event.key.keysym.sym)
            {
            case SDLK_ESCAPE:
                continuer = 0;
                break;
            default:
                    break;
            }
            SDL_Flip(ecran);
        }
    }

    TTF_Quit();
    SDL_Quit();

    return EXIT_SUCCESS;
}

int piocherMot(char *motPioche)
{
    FILE* dico = NULL;
    int nombreMots = 0, numMotChoisi = 0;
    int caractereLu = 0;
    dico = fopen("sources/dico.txt", "r");
    if (dico == NULL)   //
    {
        return 0;
    }
    do
    {
        caractereLu = fgetc(dico);
        if (caractereLu == '\n')
            nombreMots++;
    }
    while (caractereLu != EOF);
    numMotChoisi = nombreAleatoire(nombreMots);
    rewind(dico);
```

```
        while (numMotChoisi > 0)
        {
            caractereLu = fgetc(dico);
            if (caractereLu == '\n')
                numMotChoisi--;
        }
        fgets(motPioche, 100, dico);
        motPioche[strlen(motPioche) - 1] = '\0';
        fclose(dico);
        return 1;
    }

    int nombreAleatoire(int nombreMax)
    {
        return (rand() % nombreMax);
    }
```

Ce code nous permet donc d'ouvrir une fenêtre et de la refermer en appuyant sur ECHAP. Rien de bien passionnant mais ce code va nous servir de base.
Avant de commencer, il va falloir que vous téléchargiez le même pack d'images que moi.

Télécharger le pack d'image Images_Dico_Polices.zip.

Quelques variables

Premièrement, on crée un define et une énumération anonyme qui vont nous être utiles dans la suite du code.

Code : C

```
#define NOMBRE_ESSAIS 7

enum {JEU_TERMINE, JEU_EN_COURS}; // Permet de savoir si on est en
train de jouer ou non
enum {OUI, NON};   // Permet d'éviter de mettre des 0 et des 1
```

Maintenant, on va compléter le code en rajoutant des variables. Je vais donner toutes les variables d'un coup de façon à ne pas tout le temps remonter jusqu'au début du code pour définir un petit nombre et redescendre encore pour mettre un petit bout de code.
Certaines variables ne vont peut-être pas être tout à fait comprises au début mais vous verrez à quoi elles servent lorsque nous les utiliserons.

Code : C

```
SDL_Surface *ecran = NULL, *img = NULL; // On créé 2 surfaces pour
le texte et pour l'image

    SDL_Event event;    // Evenement
    int continuer = 1;
    srand(time(NULL));  // pour le nombre aléatoire

    char lettrePropose = '\b'; // on va stocker ici la lettre tapée
    char motMystere[50]; // Le mot mystère à découvrir
    if ( piocherMot(motMystere) == 0 )  // on demande un mot qui
sera mis dans motMystere
        return EXIT_FAILURE; // si on n'a pas réussi à charger le
dictionnaire on arrête le programme

    const int tailleMot = strlen(motMystere);
    char *motMystereEnCours = NULL;    // Allocation dynamique de
la mémoire
    motMystereEnCours = (char *) malloc(tailleMot * sizeof(char));
```

```
    if (motMystereEnCours == NULL)  // Verification de l'allocation
de mémoire
        return EXIT_FAILURE;  // On arrête tout

    strcpy(motMystereEnCours, motMystere);      // on copie le mot
mystère dans une autre chaine
    char tableauDeSaugegardeDesLettresTape[] = " " ;   // 26
caractères on retient les lettres que le joueur a tapé

    int nombreEssais = NOMBRE_ESSAIS;
    int reussi = JEU_EN_COURS;  // Permet de savoir si le joueur à
découvert le mot, on pourra aussi lui demander si il veut
recommencer une autre partie
    int nombreEssaisRajoute = 0; // Cette variable a un rapport
avec le nombre d'essais.
    int sauterUnTour = NON;  // Cette variable permet de sauter le
tour du joueur si il tape une lettre une seconde fois
    int numeroDeLaLettre = 0; // Le numéro de la lettre permet de
dire que A = 1 , B = 2 , . . . Elle est utilisée pour afficher les
lettres que le joueur a déjà tapé
    int i = 0;
```

Pour les noms de variables, j'ai essayé d'être le plus clair possible. Il y a de multiples possibilités pour coder ce programme. Personnellement, j'ai choisi de créer deux chaines de caractères. L'une contient le mot à découvrir (motMystere) et l'autre est la copie codée de ce mot.

C'est-à-dire qu'au début de la partie, elle sera composée de *. L'avantage de cette technique est qu'on ne doit pas s'embarrasser de ce qu'on affiche. Dans la suite, on modifiera cette chaine de caractères puis on l'affichera simplement à l'écran.

Petit remarque, j'ai fixé la longueur des tableaux à 51 caractères car l'un des mots le plus long en français est hexakosioihexekontahexaphobie (peur du nombre 666). Celui-ci ne comporte **que** 29 lettres. Vous avez aussi dans le même genre le mot hippopotomonstrosesquippedaliophobie (peur des mots trop longs). Celui-ci totalise 35 lettres. Bref, tout cela pour dire qu'il nous restera encore de la place dans le tableau.

- La surface img va permettre d'afficher les différentes images à droite.
- lettrePropose va contenir la lettre que le joueur tape. On l'initialise simplement grâce à '\b'.
- motMystere est le mot secret à découvrir.
- Les deux lignes suivantes permettent d'utiliser la fonction piocherMot. Elle arrêtera le programme si on n'a pas réussi à charger le dictionnaire.
- Le tableau motMystereEnCours est une copie de motMystere. C'est lui qui sera codé et affiché tout au long de la partie.
- Ensuite, on copie le mot mystère dans l'autre chaine et on crée un int qui va retenir la taille du mot mystère.
- Le tableau tableauDeSaugegardeDesLettresTape permet l'affichage des lettres tapées en bas de l'écran. Il permet aussi de gérer la répétition d'une même lettre.

Maintenant, on a toutes les variables pour créer notre pendu. Si vous ne comprenez pas le sens de certaines variables, ce n'est pas grave. Sachez simplement que nous les avons créées. Vous comprendrez leurs utilitées lorsque nous les utiliserons.

Initialisation

On va maintenant initialiser la SDL et SDL_ttf. Nous aurons besoin de trois polices (affichage du mot mystère, affichage des caractères, affichage de la phrase à la fin du jeu), d'une couleur noire, et de deux positions (image et texte).
C'est parti !

Code : C

```
SDL_Init(SDL_INIT_VIDEO);
ecran             = SDL_SetVideoMode(750, 320, 32, SDL_HWSURFACE |
SDL_DOUBLEBUF);

TTF_Init();
```

```
    TTF_Font *police  = NULL, *policePourFinal = NULL,
    *policePourTableau;    // création de 3 polices
    police            = TTF_OpenFont("sources/police.ttf", 45);
    policePourTableau = TTF_OpenFont("sources/policeRejouer.ttf", 25);
    policePourFinal   = TTF_OpenFont("sources/policePourTableau.ttf",
    65);

    SDL_Color couleurNoire = {0, 0, 0}; // création d'une couleur

    SDL_Rect positionTexte, positionImage;    // création de 2 positions
    positionTexte.x = 0;
    positionTexte.y = 0;

    positionImage.x = 400;
    positionImage.y = 0;

    SDL_WM_SetCaption("Pendu en SDL pour les Zeros", NULL);
```

Avant de pouvoir faire le switch pour gérer les événements, il faut initialiser motMystereEnCours. On va donc remplir le tableau d'astérisques.

Code : C

```
    for ( i = 0 ; i < tailleMot ; i++ )    // on code le motMystere
    {
        motMystereEnCours[i] = '*';
    }
```

Notre motMystere est désormais codé. Maintenant place au switch.

Gestions des événements

Ici, on va devoir faire un switch qui doit prendre en compte toutes les lettres de l'alphabet. Mais au lieu de créer un événement pour chaque lettre, on va simplement mettre cette expression : lettrePropose = event.key.keysym.sym. Cela nous permet simplement de sauvegarder la lettre.

Code : C

```
    while (continuer)
        {
            SDL_WaitEvent(&event);
            switch (event.type)
            {
            case SDL_QUIT:
                continuer = 0;
                break;
            case SDL_KEYDOWN:
                switch (event.key.keysym.sym)
                {
                case SDLK_ESCAPE:
                    continuer = 0;
                    break;
                default :
                    lettrePropose = event.key.keysym.sym;
                    switch (lettrePropose)    // Transformation clavier
QWERTY -> AZERTY
                    {
                        case 'q' :
                            lettrePropose = 'a';
                            break;
                        case ';' :
```

```
                        lettrePropose ='m';
                        break;
                    case 'z' :
                        lettrePropose ='w';
                        break;
                    case 'a' :
                        lettrePropose = 'q';
                        break;
                    case 'w' :
                        lettrePropose ='z';
                        break;
                }
                lettrePropose = toupper(lettrePropose);  //majuscule
                break;
            }
        }
```

Pourquoi utilise-t-on un "\b" ?

Simplement parce qu'il fallait choisir un caractère qui n'est pas une lettre car sinon, on va retirer un essai alors qu'il n'aura pas touché une lettre. D'ailleurs, faisons cela tout de suite. Il suffit de rajouter une condition pour ignorer le backspace.

Code : C

```
if ( lettrePropose >= 'A' &&  lettrePropose <= 'Z' || lettrePropose
== '\b')    // On annule la touche si ce n'est pas une lettre qui
est tapée
{
```

Nous avons le droit de réaliser cela car les lettres ne sont que des nombres. Peut importe leurs numéros, il suffit simplement de dire que nous voulons les lettres comprises en A et Z, soit tout l'alphabet des lettres majuscules.
Tout notre code sera placé dans cette condition. D'ailleurs, autant terminer tout de suite notre programme. Il suffit simplement de fermer la SDL et SDL_ttf à la sortie du switch.
Rajoutons la fin du programme : on libère la mémoire et on arrête tout.

Code : C

```
TTF_CloseFont(police);    // on vide la mémoire
    TTF_CloseFont(policePourFinal);
    TTF_CloseFont(policePourTableau);
    TTF_Quit();
    SDL_FreeSurface(img);
    SDL_Quit();

    return EXIT_SUCCESS;
}
```

Je vous redonne le code en entier.

Secret (cliquez pour afficher)

Code : C

```
#include <stdlib.h>
#include <stdio.h>
```

```
#include <string.h>
#include <SDL/SDL.h>
#include <time.h>
#include <SDL_image.h>
#include <SDL_ttf.h>

#define NOMBRE_ESSAIS 7

enum {JEU_TERMINE, JEU_EN_COURS}; // Permet de savoir si on est
en train de jouer ou non
enum {OUI, NON};   // Permet d'éviter de mettre des 0 et des 1
//-----------------------------------------------------------------
-----

int piocherMot(char *motPioche);
int nombreAleatoire(int nombreMax);

int main(int argc, char *argv[])
{
    SDL_Surface *ecran = NULL, *img = NULL; // On créé 2 surfaces
pour le texte et pour l'image

    SDL_Event event;    // Evenement
    int continuer = 1;
    srand(time(NULL));  // pour le nombre aléatoire

    char lettrePropose = '\b'; // on va stocker ici la lettre
tapée
    char motMystere[50]; // Le mot mystère à découvrir
    if ( piocherMot(motMystere) == 0 )  // on demande un mot qui
sera mis dans motMystere
        return EXIT_FAILURE; // si on n'a pas réussi à charger le
dictionnaire on arrête le programme

    const int tailleMot = strlen(motMystere);
    char *motMystereEnCours = NULL;    // Allocation dynamique de
la mémoire
    motMystereEnCours = (char *) malloc(tailleMot * sizeof(char));
    if (motMystereEnCours == NULL)  // Verification de
l'allocation de mémoire
        return EXIT_FAILURE;  // On arrête tout

    strcpy(motMystereEnCours, motMystere);      // on copie le
mot mystère dans une autre chaine
    char tableauDeSaugegardeDesLettresTape[] = " " ;   // 26
caractères on retient les lettres que le joueur a tapé

    int nombreEssais = NOMBRE_ESSAIS;
    int reussi = JEU_EN_COURS;  // Permet de savoir si le joueur
à découvert le mot, on pourra aussi lui demander si il veut
recommencer une autre partie
    int nombreEssaisRajoute = 0; // Cette variable a un rapport
avec le nombre d'essais.
    int sauterUnTour = NON;  // Cette variable permet de sauter
le tour du joueur si il tape une lettre une seconde fois
    int numeroDeLaLettre = 0; // Le numéro de la lettre permet de
dire que A = 1 , B = 2 , . . . Elle est utilisée pour afficher
les lettres que le joueur a déjà tapé
    int i = 0;

    SDL_Init(SDL_INIT_VIDEO);
    ecran             = SDL_SetVideoMode(750, 320, 32,
SDL_HWSURFACE | SDL_DOUBLEBUF);

    TTF_Init();
    TTF_Font *police  = NULL, *policePourFinal = NULL,
*policePourTableau;    // création de 3 polices
    police            = TTF_OpenFont("sources/police.ttf", 45);
    policePourTableau = TTF_OpenFont("sources/policeRejouer.ttf",
25);
```

```
    policePourFinal   =
TTF_OpenFont("sources/policePourTableau.ttf", 65);

    SDL_Color couleurNoire = {0, 0, 0}; // création d'une couleur

    SDL_Rect positionTexte, positionImage;    // création de 2
positions
    positionTexte.x = 0;
    positionTexte.y = 0;

    positionImage.x = 400;
    positionImage.y = 0;

    SDL_WM_SetCaption("Pendu en SDL pour les Zeros", NULL);

    for ( i = 0 ; i < tailleMot ; i++ )    // on code le
motMystere
    {
        motMystereEnCours[i] = '*';
    }

    while (continuer)
    {
        SDL_WaitEvent(&event);
        switch (event.type)
        {
        case SDL_QUIT:
            continuer = 0;
            break;
        case SDL_KEYDOWN:
            switch (event.key.keysym.sym)
            {
            case SDLK_ESCAPE:
                continuer = 0;
                break;
            default :
                lettrePropose = event.key.keysym.sym;
                switch (lettrePropose)
                {
                case 'q' :
                    lettrePropose = 'a';
                    break;
                case ';' :
                    lettrePropose ='m';
                    break;
                case 'z' :
                    lettrePropose ='w';
                    break;
                case 'a' :
                    lettrePropose = 'q';
                    break;
                case 'w' :
                    lettrePropose ='z';
                    break;
                }
                lettrePropose = toupper(lettrePropose);
//majuscule
                break;
            }

            if ( lettrePropose >= 'A' &&  lettrePropose <= 'Z' ||
lettrePropose == '\b')    // On annule la touche si ce n'est pas
une lettre qui est tapée
            {

            }


            SDL_Flip(ecran);
        }
```

```
        }

        TTF_CloseFont(police);     // on vide la mémoire
        TTF_CloseFont(policePourFinal);
        TTF_CloseFont(policePourTableau);
        TTF_Quit();
        SDL_FreeSurface(img);
        free(motMystereEnCours);
        SDL_Quit();

        return EXIT_SUCCESS;
    }

    int piocherMot(char *motPioche)
    {
        FILE* dico = NULL;
        int nombreMots = 0, numMotChoisi = 0;
        int caractereLu = 0;
        dico = fopen("sources/dico.txt", "r");
        if (dico == NULL)   //
        {
            return 0;
        }
        do
        {
            caractereLu = fgetc(dico);
            if (caractereLu == '\n')
                nombreMots++;
        }
        while (caractereLu != EOF);
        numMotChoisi = nombreAleatoire(nombreMots);
        rewind(dico);
        while (numMotChoisi > 0)
        {
            caractereLu = fgetc(dico);
            if (caractereLu == '\n')
                numMotChoisi--;
        }
        fgets(motPioche, 100, dico);
        motPioche[strlen(motPioche) - 1] = '\0';
        fclose(dico);
        return 1;
    }

    int nombreAleatoire(int nombreMax)
    {
        return (rand() % nombreMax);
    }
```

Voila, maintenant il ne reste plus qu'à mettre le code dans le if juste avant la fin du switch. Courage !

Répétition d'une même lettre

Maintenant, on commence à écrire dans le if juste avant la fin du switch

Dès le début, on doit définir si le joueur a bien tapé une lettre qu'il n'avait pas déjà proposé. On va donc avoir besoin du tableau qui retient tous les caractères tapés et vérifier si c'est bien la première fois qu'il tape cette lettre.
Si oui, alors on peut continuer. Si non, alors il faut lui faire sauter le tour tout comme on fait sauter son tour s'il ne tape pas une lettre mais une autre touche. En effet, si on ne fait pas ça, le programme va lui retirer un essai car le joueur n'aura évidemment pas trouvé de nouvelles lettres puisqu'elles sont déjà découvertes.

Code : C

```
if ( lettrePropose >= 'A' &&  lettrePropose <= 'Z' || lettrePropose
== '\b')    // On annule la touche si ce n'est pas une lettre qui
est tapée
{
     sauterUnTour = NON; // on initialise la variable
     for ( i = 0 ; i < 26 ; i++ ) // on parcourt entièrement le
tableau qui sauvegarde les lettres
     {
          if ( tableauDeSaugegardeDesLettresTape[i] == lettrePropose
&& lettrePropose != '\b') // Si on a trouvé la lettre que le joueur
a tapé
          {
               sauterUnTour = OUI; // on ne joue pas
          }
     }
```

Il suffit donc de rajouter une autre condition pour que le programme attende une autre lettre sans exécuter la suite du programme.

Code : C

```
if ( sauterUnTour == NON) // On peut continuer le programme
{
// Suite du code ici
}
```

On retire un essai

Maintenant qu'on sait que le joueur a bien tenté une lettre qu'il n'a pas déjà essayée, on peut donc lui retirer un essai. Sinon ce serait trop facile.

Code : C

```
if ( lettrePropose != '\b')  // Le joueur tente une lettre, on lui
retire son essai
     nombreEssais--;
```

Je ne retire pas systématiquement un essai car au début de la partie, la lettre a été initialisée par un backspace. Si on ne fait pas cela, au début de la partie, le joueur perdra automatiquement un essai avant même qu'il propose une lettre.

Mais si le joueur trouve une lettre, on ne doit pas lui enlever son essai, non ?

Tout à fait. La prochaine partie du code sera de remettre l'essai au joueur s'il a effectivement trouvé une lettre.

Si le joueur a trouvé un lettre, on rajoute un essai

Pour cela, nous allons, dans le même temps, modifier le mot qui s'affiche à l'écran afin de faire apparaitre les lettres trouvées. Nous allons utiliser les deux tableaux motMystere et motMystereEnCours et lui rajouter un essai si le joueur a bien trouvé une lettre.

Code : C

```
for ( i = 0 ; i <= tailleMot ; i++ ) // On parcourt l'ensemble des
lettres du mot mystère
```

```
{
     if ( motMystere[i] == lettrePropose ) // Si on a une
similitude
     {
          motMystereEnCours[i] = lettrePropose; // On change le
tableau que l'on va afficher
          if ( nombreEssaisRajoute == 0 )
          {
               nombreEssais++; // on lui remet son essai
               nombreEssaisRajoute++;    // on empêche de rajouter
autant d'essais qu'il y a de similitudes
          }
     }
}
nombreEssaisRajoute = 0;// le for est fini, on réinitialise la
variable
```

Quelques explications s'imposent. Si une lettre a été trouvée, on modifie motMystereEnCours pour que le joueur voit s'il a trouvé des lettres ou non.
La suite est peut-être un peu tordue mais c'est la seule solution que j'ai trouvé. Cela permet de ne rajouter qu'un seul essai. Si cette condition n'était pas là, le joueur gagnerait autant d'essais que de lettres trouvées ce qui serait impossible. Imaginer que le joueur termine le pendu avec encore 12 essais alors qu'il n'en a que 7 au début de la partie  .

On blitte différentes images

Maintenant que le nombre d'essais restant est connu, on peut savoir quelle image devra être blitter.

Code : C

```
switch(nombreEssais)  // On blitte des images différentes en
fonctions du nombre d'essais
{
     case 7 : img = IMG_Load("sources/0.jpg");  // En pleine forme
     case 6 : img = IMG_Load("sources/1.jpg");
     case 5 : img = IMG_Load("sources/2.jpg");
     case 4 : img = IMG_Load("sources/3.jpg");
     case 3 : img = IMG_Load("sources/4.jpg");
     case 2 : img = IMG_Load("sources/5.jpg");
     case 1 : img = IMG_Load("sources/6.jpg");  // attention, ici
     default: img = IMG_Load("sources/7.jpg");  // et pendu !
}
```

Le nombre d'essais va donc définir l'image à blitter. On a donc fini la partie image à droite. Il ne reste plus que la partie de gauche.

Le joueur a-t-il gagné ?

Avant de tout blitter, il reste plus qu'une toute petite chose à faire pour savoir si le joueur a trouvé le mot ou non. De plus, on doit sauvegarde la lettre du joueur dans le tableauDeSaugegardeDesLettresTape.

Code : C

```
reussi = JEU_TERMINE; // on initialise la variable
for ( i = 0 ; i<= tailleMot ; i++ ) // on prend le mot secret
lettre par lettre
{
    if ( motMystereEnCours[i] == motMystere[i] ||
motMystereEnCours[i] == '\b');  // Si on a trouvé une lettre else
    reussi = JEU_EN_COURS;
}
```

```
        if ( lettrePropose != '\b' )
        tableauDeSaugegardeDesLettresTape[lettrePropose-'A'] =
        lettrePropose; // on sauvegarde les lettres à son emplacement
        alphabétique
```

- Avant le for, on initialise la variable reussi avec **JEU_TERMINE**. S'il n'y a aucun changement dans la suite, le jeu sera effectivement terminé
- Le for va analysé le mot mystère lettre après lettre, s'il y a au moins une lettre qui ne va pas, on va mettre la variable reussi en **JEU_EN_COURS**. La partie n'est donc pas terminée.

Grâce à ce for, on peut savoir si on a trouvé le mot secret ou non.

On blitte tout !

Il ne reste plus qu'à blitter le tout et nous obtiendrons un pendu un peu rudimentaire, mais un pendu.

Code : C

```
        SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 210, 210, 255));
        // on efface

             SDL_BlitSurface(img, NULL, ecran, &positionImage); // on blitte
        l'image
             positionTexte.x = 50;
             positionTexte.y = ecran->h/2;
             SDL_BlitSurface(TTF_RenderText_Blended(police,
        motMystereEnCours, couleurNoire), NULL, ecran, &positionTexte); //
        on blitte le mot secret
             positionTexte.x = 10;
             positionTexte.y = 275;
             SDL_BlitSurface(TTF_RenderText_Blended(policePourTableau,
        tableauDeSaugegardeDesLettresTape, couleurNoire), NULL, ecran,
        &positionTexte); // on blitte les caractères déjà tapés
```

Les deux derniers blittages peuvent être décomposés en deux parties. On blitte une surface sur l'écran à positionTexte. Mais comme cette surface est un texte, on va directement mettre le TTF_Render_Blended dans le premier paramètre de SDL_BlitSurface. Le compilateur le comprendra très bien.

Bien sur, on n'oublie pas de refermer toutes les accolades ouvertes et de mettre un SDL_Flip(ecran) sinon vous ne verrez rien  ...

Code : C

```
        }
                  SDL_Flip(ecran);
             }
        }
```

Notre pendu est fini

Je vous donne le code en entier.

Secret (cliquez pour afficher)

Code : C

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <SDL/SDL.h>
#include <time.h>
#include <SDL_image.h>
#include <SDL_ttf.h>

#define NOMBRE_ESSAIS 7

enum {JEU_TERMINE, JEU_EN_COURS};
enum {OUI, NON};
//-------------------------------------

int piocherMot(char *motPioche);
int nombreAleatoire(int nombreMax);

int main(int argc, char *argv[])
{
    SDL_Surface *ecran = NULL, *img = NULL; // On créé 2 surfaces pour le texte et pour l'image

    SDL_Event event;    // Evenement
    int continuer = 1;
    srand(time(NULL));  // pour le nombre aléatoire

    char lettrePropose = '\b'; // on va stocker ici la lettre tapée
    char motMystere[50]; // Le mot mystère à découvrir
    if ( piocherMot(motMystere) == 0 )  // on demande un mot qui sera mis dans motMystere
        return EXIT_FAILURE; // si on n'a pas réussi à charger le dictionnaire on arrête le programme

    const int tailleMot = strlen(motMystere);
    char *motMystereEnCours = NULL;    // Allocation dynamique de la mémoire
    motMystereEnCours = (char *) malloc(tailleMot * sizeof(char));
    if (motMystereEnCours == NULL)  // Verification de l'allocation de memoire
        return EXIT_FAILURE;  // On arrete tout

    strcpy(motMystereEnCours, motMystere);      // on copie le mot mystère dans une autre chaine
    char tableauDeSaugegardeDesLettresTape[] = " " ;   // 26 caractères on retient les lettres que le joueur a tapé

    int nombreEssais = NOMBRE_ESSAIS;
    int reussi = JEU_EN_COURS;  // Permet de savoir si le joueur à découvert le mot, on pourra aussi lui demander si il veut recommencer une autre partie
    int nombreEssaisRajoute = 0; // Cette variable a un rapport avec le nombre d'essais.
    int sauterUnTour = NON;  // Cette variable permet de sauter le tour du joueur si il tape une lettre une seconde fois
    int i = 0;

    SDL_Init(SDL_INIT_VIDEO);
    ecran             = SDL_SetVideoMode(750, 320, 32, SDL_HWSURFACE | SDL_DOUBLEBUF);

    TTF_Init();  // initialisation de la SDL
    TTF_Font *police = NULL, *policePourFinal = NULL, *policePourTableau; // création de 3 polices
    SDL_Color couleurNoire = {0, 0, 0};
    police            = TTF_OpenFont("sources/police.ttf", 45);
    policePourTableau = TTF_OpenFont("sources/policeRejouer.ttf",
```

```
25);
    policePourFinal   =
TTF_OpenFont("sources/policePourTableau.ttf", 65);

    SDL_Rect positionTexte, positionImage; // création de 2
positions
    positionTexte.x = 0;
    positionTexte.y = 0;
    positionImage.x = 400;
    positionImage.y = 0;

    SDL_WM_SetCaption("Pendu en SDL pour les Zeros", NULL);

    for ( i = 0 ; i < tailleMot ; i++ )    // on code le
motMystere
    {
        motMystereEnCours[i] = '*';
    }

    while (continuer)
    {
        SDL_WaitEvent(&event);
        switch (event.type)
        {
        case SDL_QUIT:
            continuer = 0;
            break;
        case SDL_KEYDOWN:
            switch (event.key.keysym.sym)
            {
            case SDLK_ESCAPE:
                continuer = 0;
                break;
            default :
                lettrePropose = event.key.keysym.sym;
                switch (lettrePropose)
                {
                    case 'q' :
                        lettrePropose = 'a';
                        break;
                    case ';' :
                        lettrePropose ='m';
                        break;
                    case 'z' :
                        lettrePropose ='w';
                        break;
                    case 'a' :
                        lettrePropose = 'q';
                        break;
                    case 'w' :
                        lettrePropose ='z';
                        break;
                }
                lettrePropose = toupper(lettrePropose);
//majuscule
                break;
            }
        }

//-----------------------------------------------------------------
---------------------------------
        if ( lettrePropose >= 'A' &&  lettrePropose <= 'Z' ||
lettrePropose == '\b')    // On annule la touche si ce n'est pas
une lettre qui est tapée
        {
            sauterUnTour = NON; // on initialise la variable
            for ( i = 0 ; i < 26 ; i++ ) // on parcourt
entièrement le tableau qui sauvegarde les lettres
            {
                if ( tableauDeSaugegardeDesLettresTape[i] ==
```

```
        lettrePropose && lettrePropose != '\b') // Si on a trouvé la
        lettre que le joueur a tapé
                        {
                            sauterUnTour = OUI; // on ne joue pas
                        }
                    }

        //-------------------------------------------------------------
        ---------------------------------
                    if ( sauterUnTour == NON)   // On peut continuer le
        programme
                    {

        //-------------------------------------------------------------
        ---------------------------------
                        if ( lettrePropose != '\b')  // Le joueur tente
        une lettre, on lui retire son essai
                            nombreEssais--;

        //-------------------------------------------------------------
        ---------------------------------
                        for ( i = 0 ; i <= tailleMot ; i++ ) // On
        parcourt l'ensemble des lettres du mot mystère
                        {
                            if ( motMystere[i] == lettrePropose ) // Si on
        a une similitude
                            {
                                motMystereEnCours[i] = lettrePropose; //
        On change le tableau que l'on va afficher
                                if ( nombreEssaisRajoute == 0 )
                                {
                                    nombreEssais++;     // on lui remet
        son essai
                                    nombreEssaisRajoute++;    // on
        empêche de rajouter autant d'essais qu'il y a de similitudes
                                }
                            }
                        }
                        nombreEssaisRajoute = 0;    // le for est fini,
        on réinitialise la variable

        //-------------------------------------------------------------
        ---------------------------------
                        switch(nombreEssais)  // On blitte des images
        différentes en fonctions du nombre d'essais
                        {
                             case 7 : img = IMG_Load("sources/0.jpg");  //
        En pleine forme
                             case 6 : img = IMG_Load("sources/1.jpg");
                             case 5 : img = IMG_Load("sources/2.jpg");
                             case 4 : img = IMG_Load("sources/3.jpg");
                             case 3 : img = IMG_Load("sources/4.jpg");
                             case 2 : img = IMG_Load("sources/5.jpg");
                             case 1 : img = IMG_Load("sources/6.jpg");  //
        attention, ici
                             default: img = IMG_Load("sources/7.jpg");  //
        et pendu !
                        }

        //-------------------------------------------------------------
        ---------------------------------
                        reussi = JEU_TERMINE; // on initialise la
        variable
                        for ( i = 0 ; i<= tailleMot ; i++ ) // on prend
        le mot secret lettre par lettre
                        {
                            if ( motMystereEnCours[i] == motMystere[i] ||
        motMystereEnCours[i] == '\b');  // Si on a trouvé une lettre
                            else
                                reussi = JEU EN COURS;
```

```
                    }
                    if ( lettrePropose != '\b' )

    tableauDeSaugegardeDesLettresTape[lettrePropose-'A'] =
    lettrePropose; // on sauvegarde les lettres à son emplacement
    alphabétique

    //On blitte tout-----------------------------------------------------
    ----------------------------------
                    SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran-
    >format, 210, 210, 255)); // on efface

                    SDL_BlitSurface(img, NULL, ecran, &positionImage);
    // on blitte l'image
                    positionTexte.x = 50;
                    positionTexte.y = ecran->h/2;
                    SDL_BlitSurface(TTF_RenderText_Blended(police,
    motMystereEnCours, couleurNoire), NULL, ecran, &positionTexte); //
    on blitte le mot secret
                    positionTexte.x = 10;
                    positionTexte.y = 275;

    SDL_BlitSurface(TTF_RenderText_Blended(policePourTableau,
    tableauDeSaugegardeDesLettresTape, couleurNoire), NULL, ecran,
    &positionTexte); // on blitte les caractères déjà tapés
                }
                SDL_Flip(ecran);
            }
        }

    TTF_CloseFont(police);     // on vide la mémoire
    TTF_CloseFont(policePourFinal);
    TTF_CloseFont(policePourTableau);
    TTF_Quit();
    SDL_FreeSurface(img);
    free(motMystereEnCours);
    SDL_Quit();

    return EXIT_SUCCESS;
    }

    int piocherMot(char *motPioche)
    {
        FILE* dico = NULL;
        int nombreMots = 0, numMotChoisi = 0;
        int caractereLu = 0;
        dico = fopen("sources/dico.txt", "r");
        if (dico == NULL)   //
        {
            return 0;
        }
        do
        {
            caractereLu = fgetc(dico);
            if (caractereLu == '\n')
                nombreMots++;
        }
        while (caractereLu != EOF);
        numMotChoisi = nombreAleatoire(nombreMots);
        rewind(dico);
        while (numMotChoisi > 0)
        {
            caractereLu = fgetc(dico);
            if (caractereLu == '\n')
                numMotChoisi--;
        }
        fgets(motPioche, 100, dico);
        motPioche[strlen(motPioche) - 1] = '\0';
        fclose(dico);
        return 1;
```

```
}

int nombreAleatoire(int nombreMax)
{
    return (rand() % nombreMax);
}
```

Je l'ai bien aéré pour que vous voyiez les différentes étapes. Il y a beaucoup de façons différentes de coder ce pendu. Si vous arrivez au même résultat que moi et que vous n'avez pas du tout fait comme cela, et bien, je dirai que c'est normal .
J'ai réalisé ce code de manière à pouvoir facilement le découper en petits morceaux délimités par des lignes de tirets. Ce code est simplement un ensemble de petits morceaux. Cela permet de rajouter facilement d'autres morceaux.
Vous pouvez maintenant compiler ce code qui marche parfaitement.

Mais on ne sait pas quand on gagne ? Et on ne peut pas perdre non plus ?

En effet. Il manque encore deux petites choses.

- L'une est lorsque reussi = JEU_TERMINE. On a gagné. Si le joueur a gagné, on lui dit qu'il a gagné et on lui remet le mot qu'il a trouvé.
- Et l'autre est lorsque nombreEssais = 0. Le joueur a perdu. S'il a perdu, on dit au joueur qu'il a perdu et on lui donne la réponse.

Code : C

```
if ( reussi == JEU_TERMINE )  // Si on a gagné
                {
                    SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 210, 210, 255));
                    positionTexte.x = 10;
                    positionTexte.y = 120;

SDL_BlitSurface(TTF_RenderText_Blended(policePourFinal, "Felicitation ! Le mot etait bien :", couleurNoire), NULL, ecran, &positionTexte);
                    positionTexte.x = 50;
                    positionTexte.y = 180;
                    SDL_BlitSurface(img, NULL, ecran, &positionImage);
                    SDL_BlitSurface(TTF_RenderText_Blended(police, motMystereEnCours, couleurNoire), NULL, ecran, &positionTexte);
                }
//-------------------------------------------------------
                if ( nombreEssais == 0 )  // Si on a perdu
                {
                    SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran->format, 210, 210, 255));
                    positionTexte.x = 10;
                    positionTexte.y = 120;

SDL_BlitSurface(TTF_RenderText_Blended(policePourFinal, "Perdu ! Le mot etait : ", couleurNoire), NULL, ecran, &positionTexte);
                    positionTexte.x = 50;
                    positionTexte.y = 180;
                    SDL_BlitSurface(img, NULL, ecran, &positionImage);
                    SDL_BlitSurface(TTF_RenderText_Blended(police, motMystere, couleurNoire), NULL, ecran, &positionTexte);
                }
```

On doit donc rajouter cette partie à la fin du jeu.

Le point sur notre pendu

On a donc fini notre pendu maintenant !

Je vous donne le code en entier maintenant. J'ai juste rajouté les deux parties du dessus.

Secret (cliquez pour afficher)

Code : C

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <SDL/SDL.h>
#include <time.h>
#include <SDL_image.h>
#include <SDL_ttf.h>

#define NOMBRE_ESSAIS 7

enum {JEU_TERMINE, JEU_EN_COURS};
enum {OUI, NON};
//-------------------------------------

int piocherMot(char *motPioche);
int nombreAleatoire(int nombreMax);

int main(int argc, char *argv[])
{
    SDL_Surface *ecran = NULL, *img = NULL; // On créé 2 surfaces pour le texte et pour l'image

    SDL_Event event;    // Evenement
    int continuer = 1;
    srand(time(NULL));  // pour le nombre aléatoire

    char lettrePropose = '\b'; // on va stocker ici la lettre tapée
    char motMystere[50]; // Le mot mystère à découvrir
    if ( piocherMot(motMystere) == 0 )  // on demande un mot qui sera mis dans motMystere
        return EXIT_FAILURE; // si on n'a pas réussi à charger le dictionnaire on arrête le programme

    const int tailleMot = strlen(motMystere);
    char *motMystereEnCours = NULL;    // Allocation dynamique de la mémoire
    motMystereEnCours = (char *) malloc(tailleMot * sizeof(char));
    if (motMystereEnCours == NULL)  // Verification de l'allocation de memoire
        return EXIT_FAILURE;  // On arrete tout

    strcpy(motMystereEnCours, motMystere);      // on copie le mot mystère dans une autre chaine
    char tableauDeSaugegardeDesLettresTape[] = " " ;   // 26 caractères on retient les lettres que le joueur a tapé

    int nombreEssais = NOMBRE_ESSAIS;
    int reussi = JEU_EN_COURS;  // Permet de savoir si le joueur à découvert le mot, on pourra aussi lui demander si il veut recommencer une autre partie
```

```
    int nombreEssaisRajoute = 0; // Cette variable a un rapport
avec le nombre d'essais.
    int sauterUnTour = NON;  // Cette variable permet de sauter
le tour du joueur si il tape une lettre une seconde fois
    int i = 0;

    SDL_Init(SDL_INIT_VIDEO);
    ecran             = SDL_SetVideoMode(750, 320, 32,
SDL_HWSURFACE | SDL_DOUBLEBUF);

    TTF_Init();  // initialisation de la SDL
    TTF_Font *police = NULL, *policePourFinal = NULL,
*policePourTableau; // création de 3 polices
    SDL_Color couleurNoire = {0, 0, 0};
    police            = TTF_OpenFont("sources/police.ttf", 45);
    policePourTableau = TTF_OpenFont("sources/policeRejouer.ttf",
25);
    policePourFinal   =
TTF_OpenFont("sources/policePourTableau.ttf", 65);

    SDL_Rect positionTexte, positionImage; // création de 2
positions
    positionTexte.x = 0;
    positionTexte.y = 0;
    positionImage.x = 400;
    positionImage.y = 0;

    SDL_WM_SetCaption("Pendu en SDL pour les Zeros", NULL);

    for ( i = 0 ; i < tailleMot ; i++ )    // on code le
motMystere
    {
        motMystereEnCours[i] = '*';
    }

    while (continuer)
    {
        SDL_WaitEvent(&event);
        switch (event.type)
        {
        case SDL_QUIT:
            continuer = 0;
            break;
        case SDL_KEYDOWN:
            switch (event.key.keysym.sym)
            {
            case SDLK_ESCAPE:
                continuer = 0;
                break;
            default :
                lettrePropose = event.key.keysym.sym;
                switch (lettrePropose)
                {
                    case 'q' :
                        lettrePropose = 'a';
                        break;
                    case ';' :
                        lettrePropose ='m';
                        break;
                    case 'z' :
                        lettrePropose ='w';
                        break;
                    case 'a' :
                        lettrePropose = 'q';
                        break;
                    case 'w' :
                        lettrePropose ='z';
                        break;
                }
                lettrePropose = toupper(lettrePropose);
```

```
        //majuscule
                        break;
                    }
                }

        //-----------------------------------------------------------------
        ---------------------------------
                if ( lettrePropose >= 'A' &&  lettrePropose <= 'Z' ||
        lettrePropose == '\b')    // On annule la touche si ce n'est pas
        une lettre qui est tapée
                {
                    sauterUnTour = NON; // on initialise la variable
                    for ( i = 0 ; i < 26 ; i++ ) // on parcourt
        entièrement le tableau qui sauvegarde les lettres
                    {
                        if ( tableauDeSaugegardeDesLettresTape[i] ==
        lettrePropose && lettrePropose != '\b') // Si on a trouvé la
        lettre que le joueur a tapé
                        {
                            sauterUnTour = OUI; // on ne joue pas
                        }
                    }

        //-----------------------------------------------------------------
        ---------------------------------
                    if ( sauterUnTour == NON)   // On peut continuer le
        programme
                    {

        //-----------------------------------------------------------------
        ---------------------------------
                        if ( lettrePropose != '\b')  // Le joueur tente
        une lettre, on lui retire son essai
                            nombreEssais--;

        //-----------------------------------------------------------------
        ---------------------------------
                        for ( i = 0 ; i <= tailleMot ; i++ ) // On
        parcourt l'ensemble des lettres du mot mystère
                        {
                            if ( motMystere[i] == lettrePropose ) // Si on
        a une similitude
                            {
                                motMystereEnCours[i] = lettrePropose; //
        On change le tableau que l'on va afficher
                                if ( nombreEssaisRajoute == 0 )
                                {
                                    nombreEssais++;     // on lui remet
        son essai
                                    nombreEssaisRajoute++;    // on
        empêche de rajouter autant d'essais qu'il y a de similitudes
                                }
                            }
                        }
                        nombreEssaisRajoute = 0;    // le for est fini,
        on réinitialise la variable

        //-----------------------------------------------------------------
        ---------------------------------
                        switch(nombreEssais)  // On blitte des images
        différentes en fonctions du nombre d'essais
                        {
                             case 7 : img = IMG_Load("sources/0.jpg");  //
        En pleine forme
                            case 6 : img = IMG_Load("sources/1.jpg");
                            case 5 : img = IMG_Load("sources/2.jpg");
                            case 4 : img = IMG_Load("sources/3.jpg");
                            case 3 : img = IMG_Load("sources/4.jpg");
                            case 2 : img = IMG_Load("sources/5.jpg");
                            case 1 : img = IMG_Load("sources/6.jpg");  //
```

```
        attention, ici
                        default: img = IMG_Load("sources/7.jpg");  //
        et pendu !
                    }

        //-------------------------------------------------------------
        ---------------------------------
                    reussi = JEU_TERMINE; // on initialise la
        variable
                    for ( i = 0 ; i<= tailleMot ; i++ ) // on prend
        le mot secret lettre par lettre
                    {
                        if ( motMystereEnCours[i] == motMystere[i] ||
        motMystereEnCours[i] == '\b');  // Si on a trouvé une lettre
                        else
                            reussi = JEU_EN_COURS;
                    }
                    if ( lettrePropose != '\b' )

        tableauDeSaugegardeDesLettresTape[lettrePropose-'A'] =
        lettrePropose; // on sauvegarde les lettres à son emplacement
        alphabétique

        //On blitte tout-----------------------------------------------
        ---------------------------------
                    SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran-
        >format, 210, 210, 255)); // on efface

                    SDL_BlitSurface(img, NULL, ecran, &positionImage);
        // on blitte l'image
                    positionTexte.x = 50;
                    positionTexte.y = ecran->h/2;
                    SDL_BlitSurface(TTF_RenderText_Blended(police,
        motMystereEnCours, couleurNoire), NULL, ecran, &positionTexte); //
        on blitte le mot secret
                    positionTexte.x = 10;
                    positionTexte.y = 275;

        SDL_BlitSurface(TTF_RenderText_Blended(policePourTableau,
        tableauDeSaugegardeDesLettresTape, couleurNoire), NULL, ecran,
        &positionTexte); // on blitte les caractères déjà tapés
        //-------------------------------------------------------------
                    if ( reussi == JEU_TERMINE )  // Si on a gagné
                    {
                        SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran-
        >format, 210, 210, 255));
                        positionTexte.x = 10;
                        positionTexte.y = 120;

        SDL_BlitSurface(TTF_RenderText_Blended(policePourFinal,
        "Felicitation ! Le mot etait bien :", couleurNoire), NULL, ecran,
        &positionTexte);
                        positionTexte.x = 50;
                        positionTexte.y = 180;
                        SDL_BlitSurface(img, NULL, ecran,
        &positionImage);
                        SDL_BlitSurface(TTF_RenderText_Blended(police,
        motMystereEnCours, couleurNoire), NULL, ecran, &positionTexte);
                    }
        //-------------------------------------------------------------
                    if ( nombreEssais == 0 )  // Si on a perdu
                    {
                        SDL_FillRect(ecran, NULL, SDL_MapRGB(ecran-
        >format, 210, 210, 255));
                        positionTexte.x = 10;
                        positionTexte.y = 120;

        SDL_BlitSurface(TTF_RenderText_Blended(policePourFinal, "Perdu !
        Le mot etait : ", couleurNoire), NULL, ecran, &positionTexte);
                        positionTexte.x = 50;
```

```
                    positionTexte.y = 180;
                    SDL_BlitSurface(img, NULL, ecran,
&positionImage);
                    SDL_BlitSurface(TTF_RenderText_Blended(police,
motMystere, couleurNoire), NULL, ecran, &positionTexte);
                }

            }
            SDL_Flip(ecran);
        }
    }

TTF_CloseFont(police);     // on vide la mémoire
TTF_CloseFont(policePourFinal);
TTF_CloseFont(policePourTableau);
TTF_Quit();
SDL_FreeSurface(img);
free(motMystereEnCours);
SDL_Quit();

return EXIT_SUCCESS;
}

int piocherMot(char *motPioche)
{
    FILE* dico = NULL;
    int nombreMots = 0, numMotChoisi = 0;
    int caractereLu = 0;
    dico = fopen("sources/dico.txt", "r");
    if (dico == NULL)   //
    {
        return 0;
    }
    do
    {
        caractereLu = fgetc(dico);
        if (caractereLu == '\n')
            nombreMots++;
    }
    while (caractereLu != EOF);
    numMotChoisi = nombreAleatoire(nombreMots);
    rewind(dico);
    while (numMotChoisi > 0)
    {
        caractereLu = fgetc(dico);
        if (caractereLu == '\n')
            numMotChoisi--;
    }
    fgets(motPioche, 100, dico);
    motPioche[strlen(motPioche) - 1] = '\0';
    fclose(dico);
    return 1;
}

int nombreAleatoire(int nombreMax)
{
    return (rand() % nombreMax);
}
```

Et voila, notre pendu est maintenant complet. Et bien ce n'était pas si dur.

Incroyables, c'est ~~déjà~~ terminé ? Mais il nous reste encore des choses à faire, non ?

En effet. Il reste encore des choses à améliorer. **Mais** j'ai bien dit améliorer ! Et oui, que serait un TP sans . . . . . améliorations.

Améliorations

Maintenant que vous avez un pendu, il faudrait peut-être le rendre un peu plus attirant, non ? Alors on y va !

Améliorations utiles

Je vais vous donner une liste de petites améliorations. Nous avons créé ensemble un pendu qui fonctionne, cependant, il est un peu rudimentaire. Laissez libre cours à vos talents artistiques et améliorez ce pendu le plus possible.

- Ecrire un titre (et ne me demandez pas ce qu'on peut bien y mettre)
- Ecrire une petite phrase d'accueil comme par exemple : trouver le mot mystère.
- Lorsque la partie est finie, demander au joueur s'il veut recommencer une autre partie.
- Le pendu s'exécute tout de suite. Il serait plus beau d'avoir un menu qui permet d'avoir le choix entre différentes options comme jouer, quitter, à propos, ...
- Avoir un fond d'écran qui devient de plus en plus rouge lorsqu'on perd des essais.

Pour vous montrer que toutes ces améliorations sont possibles, vous pouvez télécharger l'exécutable, les DLL, les images, et le code source ici :

Télécharger le Pendu

Améliorations ~~inutiles~~ qui vous font pratiquer

Voici une idée que j'ai eu. Cependant, elle a besoin d'une autre partie car elle ne sert pas vraiment à grand chose . . .

Je trouve que le jeu est terminé quand le joueur a déjà trouvé **TOUS** les mots. Donc je me disais qu'il serait intéressant de supprimer dans le dictionnaire chaque mot que le joueur a réussi à trouver. Pour cela, lorsque le jeu est terminé et que le joueur a trouvé le mot, il faudrait appeler une autre fonction qui supprime la ligne dans le dictionnaire (petit conseil : sauvegarder la ligne du mot dans la fonction piocherMot pour ne pas avoir à tout relire le dictionnaire).

Les améliorations qui feront toute la différence

Dans un menu options au démarrage du programme, vous pouvez effectuer plusieurs réglages comme le choix du nombre d'essais, de la police et de la couleur du texte, des thèmes (par exemple, avoir le choix entre plusieurs ensembles d'images de pendu car ici, c'est toujours le même bonhomme qui se fait pendre).

Et pour ma dernière idée d'amélioration, je vous ai réservé la plus dure mais évidemment la plus belle amélioration : un pendu animé ! En effet, notre petit bonhomme se fait pendre trop rapidement pour moi. Ce serait plus "agréable" à regarder s'il pouvait être un peu animé. Par exemple, dès que le joueur vient de perdre un essai, on peut glisser quelques images une petite demi-seconde pour rendre l'action un peu plus fluide. Pour ce faire, je vous conseille ce tuto : Faire des animations avec sdlp_anim de pOpOp99O0.
Et voilà, fin du tuto.
J'espère qu'il vous a plu. Maintenant, au lieu d'avoir un pendu en console noir et blanc, vous avez un jooooli pendu en SDL.
A consommer sans modération.
Il est bien sûr inutile de dire que vous devez faire TOUS les mots du dictionnaire avant de créer un autre programme sinon .

Je tiens aussi à remercier ~~ma maison de disque~~ le formidable petit bonhomme qui se fait pendre pour nous ! Évidemment, sans lui, tout serait impossible .