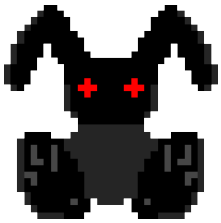

EPITECH.
L'ECOLE DE L'INNOVATION ET DE L'EXPERTISE INFORMATIQUE

Skill Test 6

Interface Dynamique

Sujet : NoCSS !

- Le Laboratoire aux Lapins Noirs -
lapinsnoirs@epitech.eu

Ce document contient un sujet d'examen associé au module d'infographie

Nom du dépôt de rendu : nocss_promotion (Exemple nocss_2042)

Index

1 – Consignes

L'examen doit être réalisé avec la LibLapin comme unique outil. Les seules fonctions autorisées sont celles précisées dans la section sujet.

La compilation sera effectuée avec les flags -Wall -Wextra.

Si votre rendu comporte un fichier binaire, un fichier .o ou un fichier tampon type « #*# » ou « *~ », vous obtiendrez la note de 1,5.

Si votre votre fonction ne s'appelle pas de la bonne façon, le programme de correction ne pourra pas la trouver et vous obtiendrez la note de 1,5.

Si votre programme est trop lent (>2 secondes), boucle à l'infinie, reçoit un signal SIGSEV, SIGFPE ou SIGPIPE, vous obtiendrez la note de 1,5.

La réussite de la compilation de votre programme avec la moulinette vous apporte la note de 2. Cette note évolue en fonction des résultats obtenus aux exercices.

Votre rendu ne doit pas comporter de main. Nous compilerons l'intégralité des fichiers .c rendus avec la moulinette. Les fichiers .h seront pris en compte si situé à la racine de votre rendu.

2 – Sujet

Préliminaire :

1 points

```
void                    tekpixel(t_bunny_pixelarray     *pix,
                                 t_bunny_position       *pos,
                                 unsigned int           color) ;
```

Écrivez la fonction suivante, qui dessine un pixel de la couleur color à la position pos dans pix.

Écrivez la fonction suivante :

1 points

```
void            tekblit(t_bunny_pixelarray          *destination,
                        const t_bunny_pixelarray    *origin,
                        const t_bunny_position      *pos) ;
```

La fonction tekblit copie la zone définie par :

origin->clipable.clip_x_position
origin->clipable.clip_y_position
origin->clipable.clip_width
origin->clipable.clip_height

à la position pos->x, pos->y, en prenant en compte les attributs origin.x et origin.y de l'image d'origine, dans l'image destination, le tout en redimensionnant correctement à l'aide des attributs scale.x et scale.y l'image.

Pour rappel :

Les attributs clip_* déterminent un sous ensemble du clipable à copier.

Les attributs scale contiennent une valeur à multiplier à la largeur et à la hauteur de la partie de l'image qu'on demande à copier.

les attributs origin.x et origin.y modifient le point d'ancrage de l'image, si par exemple, on demande à blitter un clip avec une origine a 50, 50 à la position 50, 50 dans une image, l'image sera affichée depuis la position 0, 0.

L'attribut rotation peut-être ignoré.

Écrivez la fonction suivante :

1 points

```
void            tektext(t_bunny_pixelarray          *out,
                        t_bunny_pixelarray          *fontpng,
                        const t_bunny_position      *pos,
                        const char                  *str) ;
```

Ecrivez la fonction tektext qui affiche dans out, à la position pos, la chaine de caractère str à l'aide de la police font. La position pos marque le coin en haut à gauche du texte.

Les caractères doivent être espacés d'un pixel situé sur leur droite.

Vous devez gérer le saut de ligne. En cas de texte multi-ligne, vous devez évidemment revenir à la position de départ en X avant de continuer à écrire. Chaque ligne est séparée d'un pixel.

Vous n'avez pas à gérer tabulation.

Vous pouvez ignorer les champs scale, origin, et rotation du t_bunny_clipable.

N'hésitez pas a utiliser bunny_load_pixelarray pour tester.

Écrivez la fonction suivante :

15 points

```
void                    nocss(t_bunny_pixelarray  *pix,
                              t_bunny_ini         *interface) ;
```

La fonction nocss dessine dans pix l'image décrite dans interface.
L'organisation dans interface est celle de ce fichier INI :

```
master_block=firstblock             # Quel scope représente l'image ?

[firstblock]
splittype=horizontal                # [horizontal | vertical]
splitcontent=leftblock,rightblock
backgroundcolor="0xFF808080"

[leftblock]
splittype=vertical                  # [horizontal | vertical]
splitcontent=picture,text

[picture]
backgroundpicture="lapinsnoirs.bmp"
backgrounddisplay=stretched         # [stretched | centered]

[text]
text="this is da bottom left content"
font="font.bmp"
fontsize=5,7
scale=1,1
textvdisplay=bottom                 # [top | bottom | middle]
texthdisplay=right                  # [left | right | center]
textcolor="0xFFFFFFFF"              # Color that must be applied to every
                                    # non transparent white pixels

[rightblock]
backgroundcolor="0x807777FF"        # Attention à la transparence
```

Notez l'absence de taille : tous est relatif par rapport à la taille du pixelarray !

Le scope firstblock contient par exemple un ordre signifiant qu'il faut couper horizontalement sa zone en deux (le nombre d'entrée dans splitcontent… qui aurait pu être 1, 3, 4, ou n'importe quelle autre quantité !).

La taille n'est pas nécessaire : le bloc sera coupé en deux parties égale, quelque soit la taille du bloc général.

Une boite contenant du texte doit, **si le texte est aligné en haut à gauche**, revenir à la ligne en cas de dépassement par la droite. Les alignements center et right ne **seront pas** testé sur des chaînes contenant des sauts de ligne ou dépassant de leur conteneur.

D'une manière générale, **à part pour le texte aligné à gauche**, vous n'avez pas à gérer le dépassement.

La gestion des sauts de ligne dans le texte est attendu. La gestion des tabulations n'est pas demandée. Un pixel (+Modification éventuelle de l'échelle) doit être placé entre chaque lettre, à droite et en dessous.

La profondeur des éléments du contenu est la suivante : d'abord l'image, puis la couleur de fond et enfin le texte. De plus, le bloc actuel est sous ses enfants.

Vous devez gérer la **transparence**. Les images chargées, hors police de caractère doivent néanmoins avoir, pour tous leurs pixels, la valeur 255 en transparence.

Vous avez le droit aux fonctions bunny_new_pixelarray, bunny_delete_clipable, open, close, read, lseek ainsi qu'aux fonctions INI.