Rapport de projet

Table des matières :

[Contexte du projet : 2](#_Toc469498381)

[L’écran de veille statique 3](#_Toc469498382)

[L’écran de veille dynamique 5](#_Toc469498383)

[L’écran de veille interactif 6](#_Toc469498384)

[Le lanceur 7](#_Toc469498385)

[L’historique. 8](#_Toc469498386)

# Contexte du projet :

Le but de ce projet est de créer un écran de veille pour Linux.

Plus précisément, nous devons en créer trois. Un écran de veille statique, qui affichera une image PBM (contrainte) au milieu de l’écran.

Un écran de veille dynamique, une horloge qui s’actualisera au bout de N secondes, centrée avec un message en dessous, centré aussi, avec un point qui s’ajoutera a chaque seconde d’attente.

Et enfin un écran de veille interactif, ou l’utilisateur pourra diriger un avion avec les touches H, B, G et D (que l’on a simplifié par Z, Q, S, D) et les touches 8,2,4,6. L’avion doit pouvoir traverser les murs, et commencer à s’afficher de l’autre côté de l’écran. Il doit tourner sur le point central de son fichier (Pour un fichier 6\*6, le point 3\*3 conviendra). Pour quitter l’écran de veille on doit appuyer sur ‘x’.

Toutes les images à afficher à l’écran sont des fichiers PBM, qu’il a fallu lire et afficher. Les fichiers PBM sont composer d’un nombre magique, de la taille du fichier (hauteur colonne) et du schéma en 0/1 de l’image

Tous ces écrans doivent être lancé aléatoirement depuis un lanceur. Quand on exécute ce programme, il doit choisir un seul écran, et envoyer cette donnée, ainsi que les données concernant l’écran (Quelle image choisir pour le statique par exemple) a un fichier ‘historique’.

Ce fichier doit être accessible avec la commande « -stats ». Il affichera un menu avec les options voulue.

# L’écran de veille statique

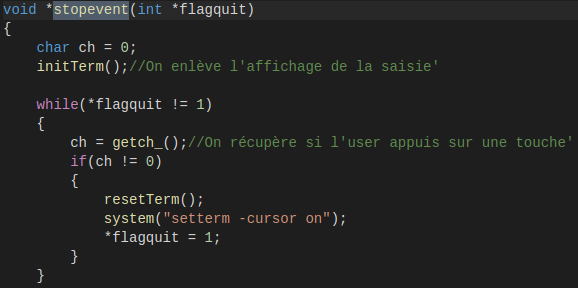
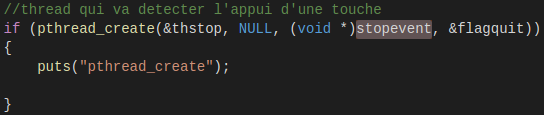
L’écran de veille statique doit afficher une image aléatoire centrée dans la console.

On a donc un dossier ou sont stocker les images PBM pour les récupérer.

Pour les charger, on va utiliser un thread :

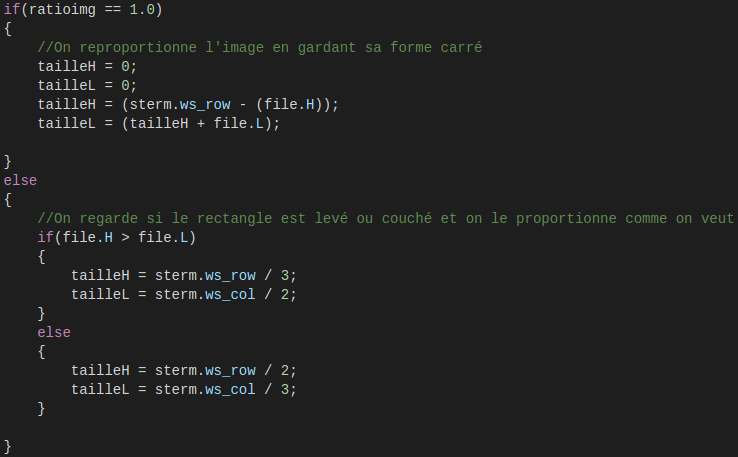


En parallèle a ce thread, il y en a un autre qui va vérifier lorsqu’on appuie sur une touche et qui va arrêter le programme si on détecte une touche :

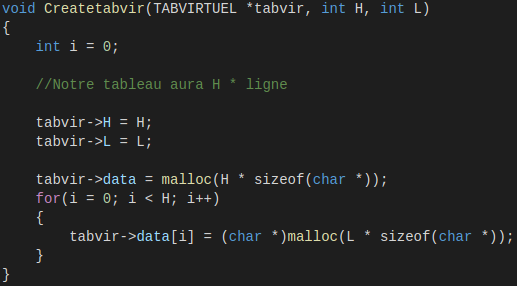


Ensuite, petit plus, on va redimensionner l’image.

On va d’abord vérifier si elle est carrée. Si elle l’est, on va juste l’adapter à la console pour qu’elle ne paraisse pas petite dans la console.

Si non, on va regarder dans quel sens est le rectangle, et adapter… l’adaptation pour que ça rende bien. Ensuite, on a plus qu’à le placer.

Pour cela on va créer un tableau virtuel ou on mettra l’image dedans.



On va logiquement mettre l’image dedans.

Après avoir fait cela, on va finalement afficher le tableau avec l’image.

# L’écran de veille dynamique

Cet écran de veille doit afficher l’heure centrée au milieu qui s’actualisera toute les N secondes. Il doit aussi afficher un message en dessous ou s’ajoutera un point toute les secondes sans actualisation.

On va donc créer en 1er les threads de lecture de fichiers PBM et de détection de touche (Les même que pour le statique).

Ensuite, on va récupérer l’heure avec une fonction, qui redonnera les unités et les dizaines dans un tableau pour savoir quels fichiers PBM récupérer.

Ensuite on redimensionne les fichiers récupérés selon la taille de la console.

On va ensuite découper le tableau en 8 parties : HH:MM:SS

On stocke dans le même temps les fichiers PBM dans leurs emplacements respectifs.

On a ensuite une boucle qui ajoute un point à la phrase DE EMRDE.

A la fin, il y a un ‘sleep’ de N temps pour réactualiser l’heure après.

# L’écran de veille interactif

Cet écran de veille affiche un avion. Pour que cet écran ait un intérêt, il faut que l’utilisateur appuie sur les touches ZQSD ou 8426 pour bouger l’avion. Une fois que l’avion a commencer à bouger, il va continuer dans la direction jusqu’à ce que l’utilisateur réappuie sur une touche de direction, ou sur la touche de sortie ‘x’. L’avion pourra évidement traverser les bords et réapparaitre de l’autre côté de l’écran.

# Le lanceur

Le but du lanceur est de… lancer. Il doit lancer un écran de veille aléatoirement, tout en envoyant son choix à l’historique. Il doit aussi choisir quelle image afficher dans le statique, et la position initiale de l’avion dans l’interactif.

# L’historique.

Le but de l’historique est d’enregistrer les choix d’écran de veille, ainsi que la date et l’heure du début de l’affichage.