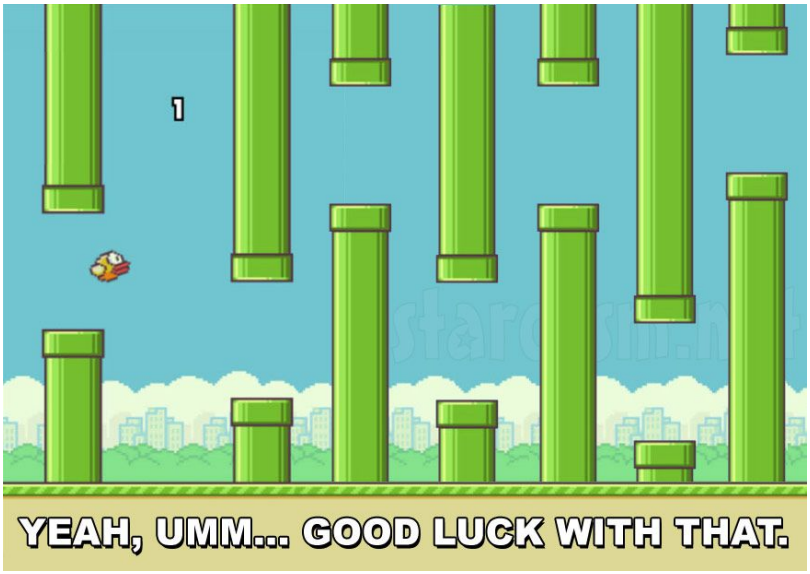
Synthèse d'Image - Projet

FLAPIMAC

L'objectif de ce projet est de vous faire réaliser un jeu d'arcade 2D, à mi chemin entre un FLAPPY BIRD et un SPACE SHOOTER à l'aide de la librairie OpenGL. Il est à réaliser en binôme.

FLAPPY BIRD



FLAPPY BIRD est un jeu mobile sorti en 2012 où le joueur contrôle un oiseau qui doit évoluer dans un niveau en évitant des obstacles ressemblant à des tubes de plomberie. L'oiseau avance tout seul et le joueur peut seulement contrôler la position verticale de l'oiseau. Réputé pour sa difficulté, le jeu officiel a été supprimé des plateformes mobiles en février 2014 suite à des accusations de plagiat et d'arnaque.

SPACE SHOOTER

Un **space shooter** est une catégorie de jeux où le joueur contrôle un vaisseau spatial qui évolue dans un monde hostile peuplé d'ennemis. Le vaisseau possède des armes permettant de détruire les ennemis. Le joueur doit aussi déplacer la position du vaisseau pour éviter les tirs ennemis.

Pour ce projet, il vous sera demandé de développer un jeu de type Space Shooter auquel s'ajoutera un mécanique de jeu d'obstacles à éviter.

1. Gameplay.

Le jeu se jouera tout seul. Le joueur incarnera un vaisseau spatial évoluant dans un monde défilant à l'horizontal. Le monde sera constitué d'obstacles à éviter, de vaisseaux ennemis qu'il faudra détruire, et de bonus à récupérer.

Les principales fonctionnalités, dans l'ordre de difficulté, qui vous seront demandées seront :

- Chargement d'un niveau depuis un fichier image au format ???
- Déplacement du vaisseau avec les touches du clavier
- Détection des collisions entre le vaisseau et les obstacles/ennemis et entre les projectiles et les vaisseaux.
- Détection de la mort du vaisseau, et de la fin du niveau.
- Modification du comportement du jeu lors de la récupération de bonus.

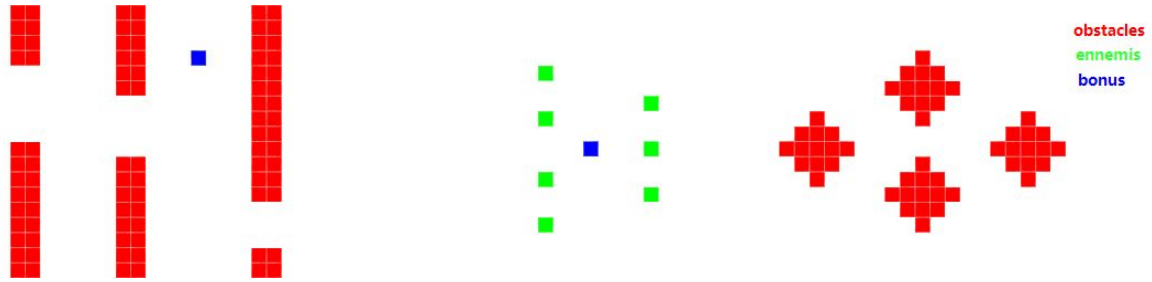
2. Chargement d'un niveau.

Chaque niveau de votre jeu sera défini par une image au format PPM que vous devrez lire dans votre programme. Vous trouverez les spécifications du format PPM à cette adresse : http://netpbm.sourceforge.net/doc/ppm.html.

Dans cette image :

- 1 pixel correspond à 1 "bloc" de l'écran. Vous devrez décider de la taille d'un bloc dans le repère global (World) afin d'afficher tous les éléments suivant la même échelle.
- Un pixel contenant du BLANC définit le fond de l'écran (pas d'objet)
- Les obstacles, les ennemis, et les bonus/malus auront un couleur attitrée.

Exemple :



Vous devrez donc varier les couleurs des pixels pour faire apparaître des types d'objets supplémentaires.

3. Déplacement du vaisseau

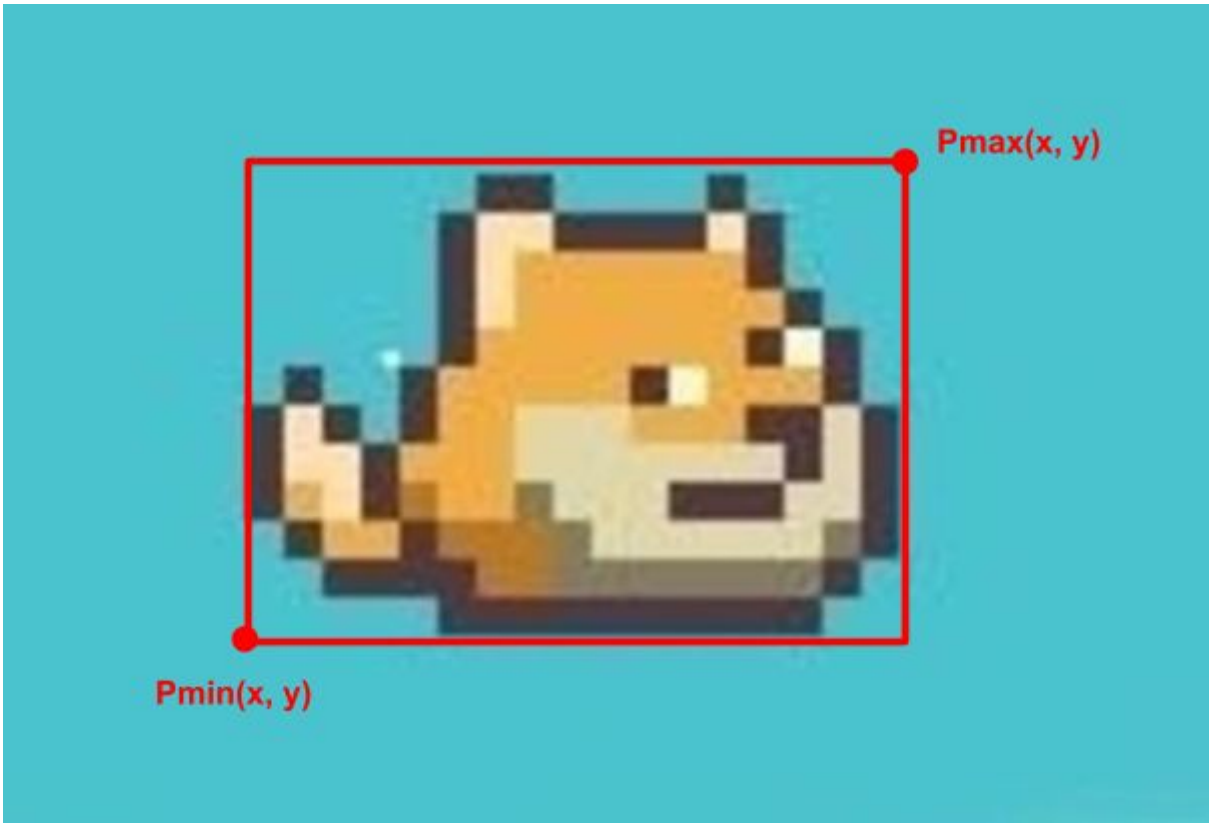
Le vaisseau sera controllable via les touches du clavier :

- Touche flèche haut : le vaisseau monte.
- Touche flèche bas : le vaisseau descend.
- Touche espace : le vaisseau tire des projectiles.

Le vaisseau devra avancer automatiquement dans le monde à une vitesse donnée. Si le vaisseau atteint le point de fin de niveau, alors le joueur a gagné et peut passer au niveau suivant. Visuellement le vaisseau sera représenté par un élément géométrique texturé.

4. Gestion des collisions

Pour gérer les collisions des objets entre eux, vous associerez à chaque objet une Bounding Box. Une bounding box est une boîte rectangulaire représentée par **un point minimum** et **un point maximum** qui représente la présence physique d'un objet. En 2D, chaque point de la Bounding Box contiendra 2 coordonnées :



Vous devrez développer un algorithme qui détecte la collision entre 2 Bounding Box.

5. Obstacles

Un obstacle est un objet statique placé dans le niveau. Si le joueur le percute, le vaisseau est détruit.

Un obstacle sera décrit par un pixel d'une certaine couleur dans votre map de niveau.

Visuellement un obstacle devra être représenté par un élément géométrique texturé.

Soyez imaginatifs dans vos créations d'obstacles. Vous n'êtes pas obligés de vous limiter à des obstacles rectangulaire. Vous pouvez ajouter des catégories supplémentaires (des obstacles circulaires, des obstacles destructibles) afin ajouter de la diversité. Attention, il vous faudra éventuellement créer plusieurs Bounding Box par obstacle pour mieux représenter les formes.

6. Ennemis

Un ennemi est un objet ressemblant à un obstacle mais qui possède des caractéristiques supplémentaires :

- Il peut se déplacer verticalement au cours du temps.
- Il peut lancer des projectiles.

Un ennemi sera décrit par un pixel d'une certaine couleur dans votre map de niveau.

Tout comme les obstacles, les ennemis seront représentés par des éléments géométriques (quad, disque...) texturés.

Vous pouvez inventer tout un tas de types d'ennemis différents : des mines (qui explosent si on tire dessus), des boss de fin de niveau, etc.

7. Projectiles

Votre vaisseau devra être capable de tirer des projectiles. Un projectile possède également une Bounding Box afin de détecter les collisions avec les ennemis.

Certains ennemis devront également avoir la capacité de tirer des projectiles que le joueur devra éviter en plus des obstacles traditionnels.

Un projectile sera également représenté par un objet texturé.

8. Fin du niveau

La fin du niveau sera décrite au choix par la détection de la fin de la map de niveau, ou par une rangée de pixels verticaux ayant une couleur donnée dans la map de niveau.
Il vous sera demandé de représenter cette fin de niveau graphiquement dans votre scène. Si le vaisseau atteint la fin du niveau, le joueur gagne la partie.

9. Modules complémentaires

En plus des fonctionnalités de base, vous pouvez développer des fonctionnalités supplémentaires. Voici une liste de fonctionnalités adaptées au jeu pour le rendre plus intéressant visuellement et en terme de gameplay..

- Utilisation de la transparence (pour les textures notamment)
- Différents types de projectiles et d'ennemis
- Ajout de bonus / malus (puissance de l'arme, augmentation/diminution de la vitesse de défilement, invincibilité, etc.). Soyez créatifs !
- Animation des "sprites" qui représentent vos objets.
- Gestion du score.
- Implémentation de plusieurs niveaux.
- Boss de fin de niveau.
- Jeu multijoueur.

10. Rendu du projet

Le rendu du projet consiste en :

1. La création d'une application de jeu qui répond aux contraintes suivantes :
    - Programmation en C.
    - Rendu avec OpenGL.
    - Fonctionne au moins sous Linux : programme qui compile avec gcc et exécutable sur les machines de la fac (typiquement celles de la salle B02).
    - Gestion de la fenêtre et des évènements avec la SDL (ou librairie équivalente type GLFW).
    - Code propre, factorisé, commenté de façon pertinente, et de préférence en anglais.

2. Un court rapport (max. 5 pages) qui indiquera : les fonctionnalités réalisées, la procédure pour compiler et lancer le jeu, l'architecture globale du jeu, les modules complémentaires réalisées ou les fonctionnalités manquantes, un screenshot de votre jeu.
3. Une soutenance qui aura lieu en fin d'année.

11. Conseils

Voici maintenant quelques conseils :

- Choisissez bien vos structures de données pour stocker les obstacles, les ennemis, les projectiles, etc. Vous devez pouvoir facilement ajouter et supprimer des objets dans ces structures de données.
- Prenez garde à la gestion du temps qui n'est pas simple. Evitez de diverger sur les modules complémentaires avant d'avoir fini les fonctionnalités principales.
- Choisissez un référentiel de coordonnées et travaillez uniquement avec celui-ci. Une possibilité serait de choisir le point (0,0) en bas à gauche du niveau mais vous faites comme vous voulez.
- Travaillez incrémentalement et ajouter les spécifications une à une (voir partie 1).
- Utilisez un logiciel de gestion de version et de travail collaboratif comme GIT. Il est fortement déconseillé de versionner son travail à la main en copiant / collant son code à différentes étapes du développement.