Equilibrium

INSTALLATION:

Le logiciel nécessite python 3.3 minimum pour fonctionner, ainsi que la librairie tkinter installé. Ensuite vous devez dé-archiver le fichier « equilibrium.zip ». Enfin il ne vous reste plus qu'a effectuer un run sur le « main.py ». Peut-être sera t-il nécessaire de modifier la première ligne de « main.py » pour y indiquer le chemin vers votre python (which python).

Le fichier « Configuration.py » permet des réglages tel que la gravité les frottements ou la taille du mobile.

FONCTIONNALITÉS:

Le logiciel a deux utilisations bien distinctes, la première est une réponse au cahiers des charges demandé soit :

- A partir d'un fichier décrivant l'arbre des poids on affiche le mobile équilibré correspondant.
- A partir d'une liste de poids, on créer un mobile équilibrée et on sauvegarde l'arbre des poids lui correspondant. La création du mobile est effectué selon 5 algorithme différents. L'un cherchant un équilibre sur les niveaux les plus profonds, un autre un équilibre futur, et enfin le dernier cherche le plus grand déséquilibre possible. Les deux autres des algorithmes esthétiques.

Le mobile créer a partir de ces algorithmes sera alors ajouté dans le dossier « Niveaux/». Il sera composé d'un numéro suivit de « arbre.txt » le numéro étant l'ordre dans lesquels les niveaux ont était créé.

La seconde est une utilisation pratiques de ces fonctionnalité dans le but de créer une activité ludique:

- Implémentation d'un moteur physique permettant de vérifier l'équilibre du mobile.
- Création d'un affichage dynamique liée a la modification des poids en temps réel.
- Système de victoire et de score liée à la qualité des action du joueur.

HYPOTHESES:

La première hypothèse méritant d'être souligné est celle liée à la création des mobiles à partir d'une liste de poids. En effet nous avons pris le parti de ne pas implémenter l'algorithme optimal d'équilibre d'un mobile à partir des poids car sa complexité était en factoriel par rapport au nombre de poids.

La seconde concerne la physique mis en place pour le jeu, en effet après un calcul de l'accélération angulaire par le principe fondamental de la dynamique, dans un soucis de pouvoir implémenter facilement d'autre contrainte au mobile tel qu'un autre pole de gravité ou des frottement, nous avons fait le choix de ne pas résoudre l'équation différentiel pour décrire le mouvement du mobile. En lieu et place nous multiplions l'accélération par un temps très petit de façon a déterminer la nouvelle vitesse ainsi que la nouvelle position, ce choix et valide si tenter que la machine qui fait tourner le logiciel dispose d'une puissance suffisante pour que le temps unitaire soit assez petit.

COMPORTEMENT INATTENDU:

Il semble exister une erreur sur les machine trop performantes dans le calcul du temps unitaire, en effet, une division par zéro a lieu. Un correctif a été apporté et l'erreur ne c'est plus produite, mais en raison du caractère imprévisible de celle-ci, rien ne

garanti qu'elle a disparu. De plus, dans un objectif de jouabilité l'arbre est prévu pour être affiché complètement sur le Canvas principal. Malheureusement, de se fait en découle un problème d'affichage : si un poids lourd est placé sur une branche profonde du mobile, celle-ci devient trop petite pour être visible. Ainsi un système de zoom et de scroll ont été implémenter, il suffit alors de modifier le fichier «Configuration.py» et la mesure 'l' pour voir apparaître les boules.

```
l = 300
h = 20
g=10
f=0.2
degrade=True
pixelTometre = 1
```

FONCTIONNALITÉS NON IMPLÉMENTÉS:

Nous aurions aimé implémenter le changement d'axe de rotation pour pouvoir comparer les comportements. En effet, nous avons fait le choix d'un axe de rotation autour du barycentre plutôt que autour de la branche soutenant les poids. De ce choix découle un comportement du mobile proche de celui du moulin, pour s'apparenter à une balance il aurait fallu faire l'autre. L'implémentation d'une telle fonctionnalité s'apparente à la multiplication des force par un sinus, cela n'a pas été implémenté car notre code n'est pas assez évolutif du point de vue de l'affichage pour permettre de changer les points fixe, et donc les position relative des différents objets. De même, pour parfaire le jeu, une ergonomie des menu plus moderne, une possibilité de sauvegarde ou de profil aurait était appréciables.

ALGORITHME:

L'algorithme AlgoMax : Son objectif est de créer le mobile le plus déséquilibré possible, ainsi, il associe de façon récursive le poids le plus lourd avec le plus léger.

```
[3, 3, 4, 5, 5, 6, 7]
[3, 4, 5, 5, 6, 10]
[4, 5, 5, 6, 13]
[5, 5, 6, 17]
[5, 6, 22]
[6, 27]
```

Ainsi pour la liste : 3,3,4,5,5,6,7

les associations se font comme ci-joint.

L'algorithme AlgoMinFutur : Son objectif est d'optimiser le mobile pour que le barycentre reste le plus centré possible et se en favorisant les barycentre les plus hauts du mobile, au détriment de ceux les plus profonds. En effet, l'algorithme par du postulat que le poids le plus faible sera toujours associé à un poids plus lourd que celui qui le succède immédiatement. En effet, les poids ne faisant que s'additionner, cela sera toujours pire.

```
[3, 3, 4, 5, 5, 6, 7]
[4, 5, 5, 6, 6, 7]
[5, 6, 6, 7, 9]
[6, 7, 9, 11]
[9, 11, 13]
[13, 20]
```

Ainsi pour la liste : 3,3,4,5,5,6,7

les associations se font comme ci-joint.

L'algorithme AlgoMinImmediat : Son objectif est d'optimiser le mobile pour que le barycentre reste le plus centré possible et se en favorisant les barycentre les plus profonds, au détriment des barycentres les plus hauts. L'algorithme trouve la différence minimum entre tout les couples poids, et créer le barycentre de ce couple minimum.

```
[3, 3, 4, 5, 5, 6, 7]
[4, 5, 5, 6, 6, 7]
[4, 6, 6, 7, 10]
[4, 7, 10, 12]
[4, 7, 22]
[22, 11]
```

Ainsi pour la liste : 3,3,4,5,5,6,7

les associations se font comme ci-joint.

L'algorithme AlgoEsth: Son objectif est de créer un mobile joli, on place un mobile binaire parfait sur l'arbre gauche et les poids restants sur l'arbre droit.

L'algorithme AlgoEsthParfait : Son objectif est de créer un mobile joli pour cela on synthétise un mobile binaire parfait auquel on ajoute sur les feuille les poids restants.

BILAN:

Les objectifs principaux du projet, à savoir : le calcul des barycentre et l'affichage du mobile à partir de la lecture du fichier source contenant un arbre ainsi que depuis un fichier contenant une liste de poids auront été remplis.

Trois algorithmes d'optimisation des mobiles avec des objectifs différents auront été implémentés.

De plus, un jeu à été développé à partir des mobiles, où le but est de dégrader les boules jusqu'à leurs disparitions tout en déséquilibrent le moins possible l'ensemble du mobile. Pour commencer ou stopper la dégradation d'une boule il suffit de cliquer sur la dite boule ou sur le bouton lui étant associé.