Ceci fait office de compte rendu pour le projet « Algorithme Génétique : Doodle Jump » du groupe Ameur/Mengozzi.

I – Résumé du projet

Le projet s’axe autour de deux composants principaux : Un jeu de type doodle jump minimaliste, jouable éventuellement par l’utilisateur, ainsi qu’un algorithme génétique. Ce dernier a pour objectif d’améliorer à chaque nouvelle partie les performances d’un joueur factice, dit « bot ».

(Lien vers la vidéo qui nous a inspiré ce projet : <https://www.youtube.com/watch?v=sB_IGstiWlc>)

II – Description du jeu

Doodle Jump est à l’origine un jeu mobile. Le joueur contrôle un petit extraterrestre qui doit s’élever de plus en plus, ce qui augmente son score. Pour cela, il doit sauter de plateformes en plateformes. Les plateformes sont rebondissantes, et le joueur ne doit s’occuper que de la direction (Droite – Gauche – Immobile). Il existe également des ennemis et des plateformes spéciales, mais dû à un planning serré, ces fonctionnalités ne seront rajoutées qu’après l’algorithme génétique, si le temps nous le permet. (Plus de détails ici : <https://fr.wikipedia.org/wiki/Doodle_Jump>).

III-/Game Functions

int random\_generator(int min, int max)

Generates an int between min and max

PLAY spawn\_player(void)

Initializes player characteristics (starting position, score, arbitrary n° of platforms, time spent mid-air and state (alive or dead))

PLAY move\_player(PLAY player, int keyboard)

Moves the player according to the key pressed and whether or not the player hits the border

void platform\_bounce(PLAY\* player, PLA\*\* platforms\_list)

Manages the jump according to the time spent mid-air and whether the player reached the platform or not

PLAY score\_up(PLAY player, PLA\*\* platforms\_list)

Manages the score of the player and the score given by the platform, if the player jumps on a platform the score is given +1 and the platform loses its ability to give anymore points (scorePlat)

PLAY death\_player(PLAY player)

If the player Y position reaches 0, the player dies

PLA\*\* malloc\_platforms\_list(void)

Allocates memory for platforms, place the first platform and calls initial\_spawn\_platform

PLA\*\* initial\_spawn\_platform(PLA\*\* platforms\_list, int index)

Creates an array of 7 platform structures and gives them a random (x,y) position but taking into account that each platform can be reached from the previous one

void scrolling(PLA\*\* platforms\_list, PLAY\* player)

When the player reaches a certain height, everything scrolls down to allow the other platforms to take place

void check\_platforms(PLA\*\* platforms\_list)

Checks if one platforms is below a certain level to allow it to be replaced

void replace\_platform(PLA\*\* platforms\_list, int previousIndex)

Changes the disappearing platform’s coordinates and scorePlat to optimize the number of platforms