Événements interprétables

Par défaut, tous les événements possèdent les mots clés suivants (tous en string):

- ***event*** : l'id de l'événement
- ***time_max*** : le temps maximum de l'événement en secondes. S'il est atteint, l'événement est stoppé, marqué comme **perdu** et le jeux passe à l'événement suivant.
- ***start_time*** : le temps en secondes au bout duquel l'événement se produit à partir de son lancement.
- une liste de conditions, encapsulées dans un *<end_conditions>*. Chaque condition représente l'état d'une variable de navette (qui se trouvent dans le fichier *state_liste.ini*). Une condition nécéssite donc deux mots clés :
    - ***key*** : l'identifiant : le nom de l'état de la navette
    - **v*alue*** : la valeur désirée. Peut-être un int, un float ou un bool

Liste des id d'événements :

- ***collision*** : événement qui simule la collision avec un objet. Joue le son d'alerte, fait trembler la navette puis affiche l'écran d'alerte
- ***wait*** : une simple fonction passive. Si *time_max* est fixée, l'événement s'arrête au bout de ce temps là.
- ***boost*** : applique un boost sur la navette. Arguments :
    - **direction** : une des directions : f, b, l, r, tp, tm, pp, pm
    - **power** : la puissance du boost. Par défaut elle vaut 1. Doit être un entier.
- ***play_sound*** : joue un son. Arguments :
    - **name :** le nom du fichier son qui doit se trouver dans le dossier des sons (sans l'extension).
- ***stop_sound****:* stoppe le son s'il est joué. Arguments :
    - **name**
- ***reset_game*** *:* remet le jeu à zéro. Réinitialise toutes les variables.
- ***start_game*** : lance le début de jeux, c'est à dire relance le scénario du début.
- ***shuttle_look_at*** : tourne la navette vers une direction fixée. Met la variable *is_moving=True* au début puis la remet à *False*. Si aucune condition ou temps maximum n'est donnée, cette étape se finit *is_moving=False* . Arguments :
    - **x**
    - **y**
    - **z**: les coordonnées de la cible
    - **time** : le temps utilisé pour se tourner en secondes. 5 par défaut.
- ***shuttle_goto*** : déplace la navette vers la direction souhaitée. Si aucune condition ou temps maximum n'est donnée, cette étape se finit *is_moving=False.* Met la variable *is_moving=True* au début puis la remet à *False*. Si aucune condition ou temps maximum n'est donnée, cette étape se finit *is_moving=False* . Arguments :
    - **x**
    - **y**
    - **z**: les coordonnées de la cible
    - **power**: la puissance du boost (fixe la vitesse). 1 par défaut.
- ***led_on*** : allume la led désirée. Arguments :
    - **id :** l'identifiant de la led. Ces leds sont listées dans le fichier state_liste.ini avec le préfixe 'l_'. Attention, certaines leds sont activées automatiquement lors d'événements. Voir ci dessous.

- ***led_off*** : éteint la led désirée.
  - **Id**
- ***update_software_state*** : met à jour un état *software* du jeu. Arguments :
  - **name** : le nom de la variable à mettre à jour. Ces variables se trouvent dans le fichier state_list.ods
  - **value :** la valeur donnée à cette variable.
- ***update_hardware_state*** : met à jour un état *hardware* du jeu, c'est à dire simule le fait d'avoir utilisé un switch, un boutton ou quoique ce soit. Cette fonction ne devrait pas être appelée en général. Arguments :
  - **name** : le nom de la variable à mettre à jour. Ces variables se trouvent dans le fichier state_list.ods
  - **value** : la valeur donnée à cette variable.
- ***control_screen_event*** : envoie un événement sur l'écran de contrôle. Arguments :
  - **name** : le nom de l'événement. Voir liste ci dessous.
  - **…** : chaque événement peut disposer d'une liste d'arguments propres.

Liste des ***control_screen_event*** *:*

- ***crew_lock_screen*** : active l'écran de verrouillage d'équipage et active le clavier. Met la variable *crew_screen_unlocked* à *True.* Si aucune condition ou temps maximum n'est donnée, cette étape se finit lorsque la variable *crew_screen_unlocked* est mise à *True.* Arguments :
  - **num_player** : le nombre de joueurs. Par défaut, il est défini dans le fichier param.ini.
- ***default_screen*** : active l'écran par défaut. Joue un son de bienvenue, stoppe les éventuelles alarmes et désactive le clavier.
- ***alert_screen*** : active l'écran d'alerte collision. Active le clavier et joue le son d'alarme en boucle jusqu'à désactivation. L'écran est déverrouillé par la liste de mots de passe dans le fichier passwords.ini sous le label *[alert_lock]*.
- ***target_screen*** : active l'écran d'entrée des coordonnées. Active le clavier. Les joueurs entrent les coordonnées d'une destination. Si elles correspondent à une position dans le fichier de codes, cette direction est utilisée.
- ***set_gauge_goto_step*** : met à jour l'incrément **par demie seconde** de la jauge désirée. Arguments :
  - **gauge** : le nom de la jauge, c'est à dire 'main_CO2', 'main_O2' ou 'main_power'
  - **step** : la valeur désirée. Quand cette jauge sera mise à jour, le curseur se déplacera de cet incrément chaque demie seconde
- ***set_gauge_goto_time***: met à jour le temps de mise à jour de la jauge désirée. Arguments :
  - **gauge** : le nom de la jauge, c'est à dire 'main_CO2', 'main_O2' ou 'main_power'
  - **time**: la valeur désirée. Quand cette jauge sera mise à jour, le curseur se déplacera de sa valeur actuelle à la nouvelle valeur en ce temps **time** (donné en secondes)
- ***update_state*** : met à jour un composant sur l'écran de bord principal. Nécessite d'être en écran *default_screen*. Arguments :
  - **key** : l'identifiant du composant. Si aucun id n'est passé, toutes les variables sont mises à jour. Ces id correspondent en fait à des variables de navette. Chaque fois qu'une variable est changée, si elle apparaît sur cet écran l'écran est mis à jour en conséquence.

Liste des variables qui apparaissent sur l'écran de contrôle

- ***batterie1/2/3/4.***
- ***offset_ps_x***

- ➔ ***offset_ps_y***
- ➔ ***recyclage_O2/CO2/H2O***
- ➔ ***tension_secteur1/2/3***
- ➔ ***oxygene_secteur1/2/3***
- ➔ ***thermique_secteur1/2/3***
- ➔ ***target_name***
- ➔ ***freq_comm***
- ➔ ***pilote_automatique***
- ➔ ***sp_power***
- ➔ ***moteur1/2/3***
- ➔ ***correction_roulis/correction_directeion/correction_stabilisation***

Effet des boutons / switch hardware.

Rappel : la première lettre désigne la nature du composant : **s** = switch, **b**=boutton, **j**=joystick, **l**=led. *Les commandes hardwares sont prioritaires*. Si le symbole ☼ est présent, ce bouton/switch est connecté à une led verte qui s'allume ou non.

Dans tous les cas, le tableau de bord est mis à jour.

- **s_moteur1** : ☼ commande l'état du moteur1
- **s_moteur2** : ☼ ” ” moteur2
- **s_moteur3** : ☼ ” ” moteur3
- **s_moteur4** : ☼ ” ” moteur4
- **s_pilote_automatique1**
- **s_pilote_automatique2** : ☼ (une led pour les deux) si les deux sont désactivés, ils désactivent le pilote automatique et toutes les corrections. S'ils sont tous deux réactivés et que ça fonctionne, les corrections *ne sont pas* réactivées**.**
- **b_correction_roulis :** ☼
- **b_correction_direction :** ☼
- **b_correction_stabilisation :** ☼ si le pilote automatique est activé, ces corrections remettent la navette dans la direction souhaitée (à définir).
- **b_freq_moins**
- **b_freq_plus** : augmente ou diminue la fréquence de communication de **50** MHz si l'écran actuel est l'écran **default_screen**
- **s_recyclage_O2** : ☼
- **s_recyclage_CO2 :** ☼
- **s_recyclage_H2O** : ☼ active ou non des recyclages (consomme de l'énergie).
- **j_sp_orientation_h** : contrôle l'orientation des panneaux solaires en horizontal.
- **j_sp_orientation_v** : contrôle l'orientation des panneaux solaires en vertical.
- **s_batteries** : ☼ si ON, permet d'activer les batteries.
- **s_batterie_1 :** ☼
- **s_batterie_2 :** ☼
- **s_batterie_3 :** ☼
- **s_batterie_4 :** ☼ active les batteries 1 à 4.

A propos de l’énergie.

Certains éléments, quand ils sont activés, fournissent ou consomment de l’énergie. Cette valeur est fournie dans le fichier power_balance.ini

A côté de chaque élément, on y lit la puissance produite (si positive) et consommée (si négative). Après le choc de la navette, c’est à dire à l’appel de l’événement ***collision***, la navette change d’état.

1. *Avant la collision* : tout fonctionne correctement. Les batteries sont désactivées et les panneaux solaires sont correctement orientés. Les trois moteurs produisent 90 W et les panneaux solaires 10 W. L’ensemble consomme donc 100W.
2. *Après la collisions* : deux moteurs sont hors d’état de fonctionner. La puissance diminue à 20W et les panneaux solaires subissent un offset. Leur puissance diminue comme 1/ (offset_x^2 + offset_y^2). Les valeurs de l’offset après la collision sont données par **hit_offset_ps_x** et **hit_offset_ps_y**.

Le but est de se ramener à une consommation normale en désactivant toutes les tâches non nécessaires pour réactiver le pilote automatique.

On doit donc avoir : 40 (prod) – 40 (pilotage) – 15 (secteur 1) + 4*batterie = 1

=> batterie = 4