Création d'un scénario

Notes :

- Dans ce manuel, le mot « instruction » correspondra à une action que le simulateur effectuera (afficher du texte, poser une question à l'utilisateur, …) et le mot « tâche » correspondra à une action que l'utilisateur devra effectuer (prendre un objet, vérifier un équipement, …).
- Dans une instruction ou une tâche, <paramètre> veut dire que <paramètre> doit être remplacé par la valeur voulue.
- Un paramètre marqué entre || signifie qu'il est facultatif.
- Ce manuel sera mis à jour lorsque de nouvelles fonctions seront ajoutées.

1) Commentaires

Les commentaires sont des lignes qui ne seront pas prises en compte par le simulateur. Ils peuvent servir à prendre des notes (pour soi ou pour les personnes amenées à travailler sur le scénario) ou désactiver temporairement des lignes à des fins de test.

Pour faire passer une ligne en commentaire, il suffit de la commencer par « # ».

2) Bloc d'instructions ou de tâches

Un bloc correspond à un ensemble de tâches ou d'instructions. Pour écrire un bloc, on commence par écrire son nom suivi de « : », puis on va la ligne, on ajoute une tabulation, puis on écrit son contenu. Le bloc se termine lorsqu'une ligne a une tabulation de moins. Autrement dit, on commence un bloc en avançant d'une colonne puis on le termine en reculant d'une colonne.

Il en existe plusieurs :

- define

```
define:
    $var1 = obj1
    $var2 = obj2
    $var3 = obj3
```

Celui-ci permet de définir des variables qui pourront être utilisées dans le reste du fichier (seulement dans les tâches « take » pour l'instant, dans d'autres tâches et instructions par la suite). La déclaration des variables se fait en écrivant $<nom variable> = <contenu variable>. Pour les utiliser, il suffit d'écrire $<nom variable> là où on souhaite l'utiliser.
Exemple : si on reprend les variables déclarées ci-dessus, « take $var1 » est équivalent à « take obj1 ».

- tasks without order/tasks with order

```
tasks without order:
    |strict take|
    take obj1
    take obj2
    take obj3
    |wait for validation|
```

Ce bloc contient les différentes tâches que l'utilisateur doit accomplir. Le bloc « tasks without order » contient des tâches pouvant être réalisées dans n'importe quel ordre. Le bloc « tasks with order » contient des tâches devant être réalisées dans l'ordre indiqué (l'ordre dans lequel elles sont écrites). Les deux blocs peuvent exister dans un même scénario.

« strict take » n'est pas une tâche (donc son emplacement dans la liste de tâches n'a pas d'importance), mais indique au simulateur que l'utilisateur doit effectuer seulement les tâches de type « take » indiquées (il ne doit prendre que les objets demandés et pas un de plus). Sans ce paramètre, il suffira à l'utilisateur d'avoir au moins les objets demandés (peu importe si il en a pris d'autres en plus).

« wait for validation » n'est pas une tâche (donc son emplacement dans la liste de tâche n'a pas d'importance), mais indique au simulateur qu'il doit attendre une validation de la part de l'utilisateur (via un bouton, par exemple) pour vérifier si toutes les tâches sont effectuées et lancer les instructions correspondant au résultat (succès ou échec). Sans ce paramètre, le simulateur vérifiera à chaque action de l'utilisateur.

- when success/when fail

```
when success:
    print "Choix d'équipement correcte !" for 3s
    load scene01b

when fail:
    print "Choix d'équipement incorrecte !" for 3s
    restart
```

Ces blocs sont appelés lorsque l'on vérifie les tâches. Si toutes les tâches sont effectuées (et dans le bon ordre si il y en a un), on lance le bloc « when success ». Sinon, on lance le bloc « when fail ».

- when firsttry

```
when firsttry:
    print "Choisissez le bon équipement" with confirmation
```

Bloc d'instructions appelé uniquement lors du premier lancement du scénario. Si on recommence le scénario (suite à un échec, par exemple), ce bloc ne sera pas appelé à nouveau.

3) Instructions

Une instruction correspond à une action que le simulateur effectuera.
Il en existe plusieurs :

- print

```
print "<texte à afficher>" with confirmation
print "<texte à afficher>" for <nombre de secondes>s
```

Permet d'afficher du texte. « with confirmation » permet d'indiquer que le simulateur doit attendre une confirmation de l'utilisateur avant d'enlever le texte et de passer à l'instruction suivante. « for <n>s » indique que le simulateur doit attendre n secondes avant d'enlever le texte et de passer à l'instruction suivante.

- load

```
load <nom du fichier>
```

Permet de lancer un autre fichier de scénario. Réinitialise également l'inventaire.

- restart

```
restart
```

Permet de relancer le scénario (annule les tâches effectuées, vide l'inventaire, …).

- checkbox

```
checkbox |strict| "<question>" "<réponse1>" "<réponse2>" "<réponse3>":
        #si réponse 1 et 2 sélectionnées
        case 1,2:
            print "Réponses 1 et 2 sélectionnées"

        #si réponse 3 sélectionnée
        case 3:
            print "Bonne réponse !" with confirmation

        #si les réponses sélectionnées ne correspondent à aucun des cas ci-dessus
        else:
            print "Mauvaise réponse !" for 3s
            #permet
            reask
```

Permet de poser une question à choix multiples à l'utilisateur. Un bloc case correspond aux instructions à effectuer si les réponses sélectionnées correspondent à celles indiquées devant case. Si plusieurs réponses doivent être sélectionnées, il suffit d'écrire les numéros séparés par des virgules.
Exemple : « case 1,2 » signifie que l'utilisateur doit sélectionner les réponses 1 et 2.

Si « strict » est indiqué après « checkbox », cela indique qu'il faut que les réponses de l'utilisateur correspondent exactement aux réponses attendues (donc sans réponse(s) en plus de celle(s) attendue(s)).
Exemple : si les réponses sélectionnées sont les réponses 1, 2 et 3 et que les réponses attendues par le bloc case sont les réponses 1 et 2 :

- Si strict est indiqué : le bloc case pour les réponses 1 et 2 ne sera pas exécuté car {1,2} ≠ {1,2,3} ({réponses attendues par le bloc} ≠ {réponses de l'utilisateur})

- Si strict n'est pas indiqué : le bloc case pour les réponses 1 et 2 sera exécuté car {1,2} ⊆ {1,2,3} ({réponses attendues par le bloc} ⊆ {réponses de l'utilisateur})

4) Exemple

```
define:
    #les valeurs correspondent au nom des objets dans le moteur Unity
    $tshirt = Shirt
    $tenueCirculation = Combinaison
    $chaussettes = Chaussettes
    $chaussuresTravail = Sole
    $dosimetreActif = DosiMActif
    $dosimetrePassif = DosiMPassif

when firsttry:
    print "Choisissez le bon équipement" with confirmation

tasks without order:
    strict take
    take $tshirt
    take $tenueCirculation
    take $chaussuresTravail
    take $dosimetreActif
    take $dosimetrePassif
    wait for validation

when success:
    print "Choix d'équipement correct !" for 3s
    load scene01b

when fail:
    print "Choix d'équipement incorrect !" for 3s
    restart
```