Présentation projet programmation système

Réalisation de mécanismes pour un jeu de plateforme (voir dépôt)

COUVY Julien

LEYX Sébastien

Objectifs

- Système de sauvegarde et de chargement des cartes
- Utilitaire de manipulation de sauvegardes
- Gestion des temporisateurs.

Partie 1: Sauvegarde et chargement

Sauvegarde écrite en binaire dans un fichier `saved.map`
Ordre d'écriture des informations:

Type d'information	Taille
Largeur carte	U_INT
Hauteur carte	U_INT
Nb d'objets max	U_INT
Nb d'objets présents	U_INT
Objet n°1	...
Matrice	L * H * U_INT

Partie 1: Sauvegarde et chargement

Structure de la sauvegarde d'un objet x

Type d'information	**Taille**	**---**	**...**	**...**
Présent dans map	INT	---	**Collectibilité**	INT
Type	INT	---	**Générateur**	INT
Frames	INT	---	**Longueur nom**	INT
Solidité	INT	---	**Nom de la texture**	longueur * CHAR
Destructibilité	INT	---		

Partie 1: Sauvegarde et chargement

Exemple sur ordinateur

Nous avons pré-enregistré une map permettant de tester les caractéristiques des objets à la sauvegarde et leur bonne conservation.

```
$> ./game -l maps/test_objets.map
```

Partie 2: Utilitaire maputils

Implémentation

Utilisation de la fonction `getopt` pour la gestion des différentes options.

```
struct option long_option[] =
{
    {"getwidth",    no_argument,          0,   GET_WIDTH},
    {"getheight",   no_argument,          0,   GET_HEIGHT}
    {"getobjects",  no_argument,          0,   GET_OBJECTS
    {"getinfo",     no_argument,          0,   GET_INFO},
    {"setwidth",    required_argument,    0,   SET_WIDTH},
    {"setheight",   required_argument,    0,   SET_HEIGHT}
    {"setobjects",  required_argument,    0,   SET_OBJECTS
    {"pruneobjects",no_argument,          0,   PRUNE_OBJ},
    {0,             0,                    0,   HELP }

};
```

Partie 2: Utilitaire maputils

Exemple d'utilisation

```
$> cd util/
$> ./maputils ../maps/saved.map --option (voir ci-dessus)
```

Notre compréhension de la fonction `setobjects` est qu'elle doit ajouter les objets passés en paramètres si ces derniers ne sont pas déjà présents, ou les modifier s'ils sont déjà là.

Partie 3: Gestion des temporisateurs

Fonctionnement global

On utilise une liste doublement chaînée d'évènements représentés par la structure `Event` :

```
typedef struct event_s {
    unsigned long daytime;
    struct itimerval delay;
    void* event_param;
    struct event_s *prev;
    struct event_s *next;
} Event;
```

Les évènements sont triés dans l'ordre chronologique.

Partie 3: Gestion des temporisateurs

Exemple d'utilisation

Nous avons réalisé un niveau contenant tous les objets présents dans le fichier `objets.txt` fourni dans le sujet.
La sauvegarde contient des mines cachées 😄.

```
$> ./game -l maps/test_fullmap.map
```