Table des matières

I – Définition

ScenesFx est un module conçut pour la gestion des chargements de scènes dans un jeu. C'est un module très puissant qui intègre un système de chargement dynamique de niveau ainsi qu'une optimisation au cours des chargements.
NB : Ce module est de nature indestructible, est compatible à un jeu 2D, 3D et est sauvegardable.

II – Les propriétés disponibles

+ int Optimization = 0 : Contient le mode d'optimisation utilisée pour la gestion des chargements automatiques des scènes d'un niveau donné. Les valeurs possibles sont :
-> **ScenesFx.Optimisation.NONE ou 0** : Aucune optimisation ne sera appliquer.
-> **ScenesFx.Optimisation.VISIBILITY ou 1** : L'optimisation ciblera la visibilité des scènes.
-> **ScenesFx.Optimisation.DESTRUCTION ou 2** : L'optimisation ciblera la destruction des scènes.

+ bool SyncLoading = true : Devons-nous attendre qu'une scène en cours de chargement, se charge entièrement avant de lancer le chargement de la prochaîne scène ?

+ Dictionary Levels : Contient tous les niveaux du jeu. Dans ce dictionaire, le développeur doit suivre la nomenclature *NomDuNiveau :{}*. Les alcolades font référence à un dictionaire représentant toutes les scènes du niveau en question. Dans ce dictionaire, on suivera la nomenclature : *NomDeLaScene : Dictionaire | String*. Lorsqu'on précise le nom de la scène à prendre en charge, on a le choix entre deux type de valeurs : Une chaîne de caractères contenant le chemin pointant vers la scène en question ou un dictionaire supportant les clés suivantes :

» **String path** : Contient le chemin pointant vers la scène à prendre en charge. De préférence, l'extension de la scène doit être le (.tscn) ou (.scn). L'utilisation de cette clé est obligatoire.

» **PoolStringArray borders** : Quelles sont les noms des scènes voisines de la scène qui a été prise en charge ? Cette section est très important car elle qui est solicité dans les chargements automatiques des scènes d'un niveau. Notez que toutes les scènes que vous configurez ont forcément des voisines. Dans la réalisation d'un jeu de haut niveau, l'optimisation des scènes est une étape incontournable car elle consiste à déterminé les scènes qui se trouvent à la porté du joueur et celles qui sont hors de sa porté. Ainsi nous pouvont décider de masquer ou détruire celles qui sont en dehors du champ de vision du joueur et d'afficher ou charger celles qui se retrouvent à la porté du joueur sans qu'il ne s'en rendre compte. Cette technique évite à la mémoire graphique d'être trop surchargé diminuant ainsi le taux de ramage du jeu. Cette méthode de chargement ressemble à l'*occlusion culling*. Si vous voulez mettre en place un tel système, vous devez faire appel à la clé *borders* contenant les noms des scènes immédiatement voisine de la scène en question. En d'autres termes, ces dernières représenteront le champ de vision ou la porté de la scène en question. Ainsi, si vous connaissez la scène où se trouve le joueur alors vous avez la porté de sa vision et par conséquant décidé quelle scène, il faut cachée ou détruire et quelle scène, il faut affichée ou chargée. Cette opération est automatique car géré par le module lui même. Notez

que les noms des scènes que vous préciserez dans cette clé doivent avoir été prises en charge dans le module.

» **int priority = 0** : Quelle est la priorité de chargement de la scène ciblée ? Notez que quelque soit la priorité, les scènes statiques sont d'abord chargées avant celles dynamiques.

NB : Les répétitions des noms des scènes et des niveaux ne sont pas tolérées. Avant de poursuivre il y a deux choses que vous devez savoirs : Le module fonctionne grâce à deux éléments de base qui sont la staticité des scènes ainsi que leur dynamicité. Il y a deux types de scènes qui sont les scènes statiques et celles dynamiques. Les scènes statiques sont chargées quelque soit la progression du joueur dans les scènes du niveau et celles dynamiques sont chargées en fonction de la progression du joueur dans les scènes du niveau. La notion de scène statique est apparut à cause de certains constats. Par exemple le skybox d'un niveau est toujours présent quelque soit la position du joueur. Le skybox pourrai donc être mis dans une scène qui lui est propre afin d'être chargé de façon statique pour qu'il soit toujours présent dans le niveau qui le solicite. La même chose pourrai être aussi faite sur le joueur du jeu. Cependant pour permettre à ce qu'une scène soit chargée en mode statique, il faut mettre le caractère "*$*" devant son chemin d'accès au cours de sa configuration. Par défaut toutes les scènes d'un niveau sont chargées. Il est possible de spécifier celles qui seront uniquement chargées dans le niveau en mettant le caractère "@" sur le chemin d'accès des scènes désignées comme étant celles à chargées par défaut au cours de leur configuration. Il est conseillé de préciser uniquement les premières scènes du niveau pour préserver l'optimisation. Gardez à l'esprit que cette précision n'est solicité que si aucune progression n'a été éffectuée dans les scènes du niveau en question. **Une scène ne peut pas être à la fois dynamique et statique. Evitez également de faire des scènes trop lourdes à chargées**.

III – Les méthodes disponibles

+ void loads (data) : Charge une ou plusieurs scène(s) ou un ou plusieurs niveau(x) du jeu.

» **Array data** : Tableau de dictionaire contenant toutes les configurations de tou(te)s les scènes ou niveaux à chargé. Ce dictionaire suporte les clés suivantes :

» **String | int id** : Contient l'identifiant du niveau ou de la scène à chargé(e). Cependant, pour chargé un niveau, l'identifiant doit contenir soit le nom du niveau en question, soit son index de position. Pour charger une scène, l'identifiant devra suivre la nomenclature suivante : *NomDuNiveau|Position/NomDeLaScene|Position*. Si le niveau n'a pas été précisé, celui actif sera pris pour cible pour faire le traitement demandé. L'utilisation de cette clé est obligatoire.

» **bool open = true** : Voulez-vous importer automatiquement la scène ou le niveau après chargement ?

» **bool borders = true** : Voulez-vous charger également les scènes voisines de la scène en question ? Cette clé ne s'utilise que lorsque l'élément à chargé est une scène et non un niveau.

» **bool reversed = false** : Voulez-vous renverser l'ordre de chargement des scènes ? Cette clé ne s'utilise que si les scènes voisines de la scène en question sont prises en charge pour le chargement.

» **int | float interval = 0.0** : Quel est l'intervalle de temps avant chaque chargement ?

» **bool destroy = false** : Souhaitez-vous détruire l'ancienne version de la scène ou du niveau en question avant un nouveau chargement ? Cette clé peut être utilisée pour éffectuer des rechargements de niveau(x) ou de scène(s). Si cette option n'est pas activée, toute scène ou tous niveau déjà chargé(e) ne peut être rechargé.

+ **void visibilities (data) :** Contrôle la visibilité d'un ou de plusieurs scène(s).
» **Array data** : Tableau de dictionaire contenant toutes les configurations sur la visibilté de toutes les scènes. Ce dictionaire suporte les clés suivantes :
» **String | int id** : Contient l'identifiant d'une scène. L'utilisation de cette clé est obligatoire.
» **bool visible = true** : Voulez-vous changer la visibilité de la scène en question ?
» **int | float interval = 0.0** : Quel est l'intervalle de temps avant chaque changement d'état ?

+ **void show_scenes_without (id, delay = 0.0) :** Rend visible toutes les scènes déjà chargées dans le niveau actif exceptées celles renseignées dans le paramètre *id*.
» **String | Array | PoolStringArray | PoolIntArray | int id** : Contient le(s) identifiant(s) de la ou des scène(s) à ciblée(s).
» **float delay** : Quel est le temps mort avant les changements d'états ?

+ **void hide_scenes_without (id, delay = 0.0) :** Rend invisible toutes les scènes déjà chargées dans le niveau actif exceptées celles renseignées dans le paramètre *id*.
» **String | Array | PoolStringArray | PoolIntArray | int id** : Contient le(s) identifiant(s) de la ou des scène(s) à ciblée(s).
» **float delay** : Quel est le temps mort avant les changements d'états ?

+ **Array | Node get_enabled_scenes () :** Renvoie les références de toutes les scènes visible.

+ **Array | Node get_disabled_scenes () :** Renvoie les références de toutes les scènes invisible.

+ **Node | Array get_scenes (id) :** Renvoie le(s) référence(s) de la ou des scène(s) données à travers leur identifiant.
» **String | Array | PoolStringArray | PoolIntArray | int id** : Contient le(s) identifiant(s) de la ou des scène(s) à ciblée(s).

+ **Array | Node get_loading_scenes () :** Renvoie les références de toutes les scènes en cours de chargement sur le niveau en question.

+ **Array | Node get_loaded_scenes () :** Renvoie les références de toutes les scènes déjà chargées sur le niveau en question.

+ **bool is_loading (id) :** Est-ce que la ou les scène(s) ou le(s) niveau(x) donné(es) sont-ils/elles en cours de chargement ? Utilisez la nomenclature précisée au niveau de la méthode *loads ()*.
  » **String | Array | PoolStringArray | PoolIntArray | int id** : Contient le(s) identifiant(s) de la ou des scène(s) ou du/des niveau(x) à ciblé(es).

+ **bool is_loaded (id) :** Est-ce que la ou les scène(s) ou le(s) niveau(x) donné(es) ont-ils/elles été déjà chargé(es) ? Utilisez la nomenclature précisée au niveau de la méthode *loads ()*.
  » **String | Array | PoolStringArray | PoolIntArray | int id** : Contient le(s) identifiant(s) de la ou des scène(s) ou du/des niveau(x) à ciblé(es).

+ **int get_progress (id) :** Renvoie la valeur de progression de la scène ou du niveau donné(e). Utilisez la nomenclature précisée au niveau de la méthode *loads ()*.
  » **String | int id** : Contient l'identifiant de la scène ou du niveau à ciblé(e).

+ **bool game_progress () :** Le joueur a t-il progressé dans les scènes du jeu ?

+ **Node get_level () :** Renvoie la référence du niveau actuellemet chargé.

+ **String get_active_scene_name (object_id) :** Renvoie le nom de la scène active par le nom de son hôte.
  » **String object_id** : Contient le nom hôte qui n'est d'autre que celui d'un objet ou d'un noeud ou encore d'un joueur par exemple. Notez que le module gère la position de plusieurs objets à la fois. Considérez l'objet comme un point représentant la position de la scène active. Ainsi à partir de la position de l'objet, on peut connaître la scène où il se trouve (scène active) et donc déterminer ces voisines (champ de vision ou porté).

+ **void set_active_scene_name (object_id, name, delay = 0.0) :** Prévient le module de la nouvelle position de l'hôte dans le niveau en question. Notez que cette méthode joue un rôle très important dans le chargement automatique des scènes d'un niveau ainsi que la progression dans les scènes du jeu.
  » **String object_id** : Contient le nom hôte qui n'est d'autre que celui d'un objet ou d'un noeud ou encore d'un joueur par exemple. Notez que le module gère la position de plusieurs objets à la fois. Considérez l'objet comme un point représentant la position de la scène active. Ainsi à partir de la position de l'objet, on peut connaître la scène où il se trouve (scène active) et donc déterminer ces voisines (champ de vision ou porté).
  » **String name** : Contient le nom de la scène où se trouve l'hôte.
  » **float delay** : Quel est le temps mort avant la mise à jour de la position de l'hôte ?

+ **String | PoolStringArray get_level_names_of (id) :** Renvoie le(s) nom(s) du/des niveau(x) contenant la ou les scène(s) donnée(s) grâce à leur identifiant.
  » **String | Array | PoolStringArray | PoolIntArray | int id** : Contient le(s) identifiant(s) de la ou des scène(s) à ciblé(es).

+ **void load_last_level (open = true, border = true, reversed = true, delay = 0.0) :** Charge le niveau récupéré dans le gestionnaire des données du jeu. En d'autres termes, si l'utilisateur sauvegarde les données issues de ce module, ce dernier pourra récupérer le niveau et la scène ou se trouvait le joueur avant qu'il ne quitte le jeu. Il est absolument obligatoire de mettre à jour le module avec la méthode *set_active_scene_name ()* pour permettre à ce dernier de savoir quel niveau et quelle scène chargée à l'appelle de la méthode *load_last_level ()*.
  » **bool open** : Voulez-vous importer automatiquement la scène ou le niveau après chargement ?
  » **bool borders** : Voulez-vous charger également les scènes voisines de la scène en question ? Cette clé ne s'utilise que lorsque l'élément à chargé est une scène et non un niveau.
  » **bool reversed** : Voulez-vous renverser l'ordre de chargement des scènes ? Cette clé ne s'utilise que si les scènes voisines de la scène en question sont prises en charge pour le chargement.
  » **float delay** : Quel est le temps mort avant le chargement ?

+ **void destroy (id = null, delay = 0.0) :** Détruit le niveau ou la/les scènes donné(es) grâce à leur identifiant. Par défaut, le niveau actif est pris pour cible.
  » **String | Array | PoolStringArray | PoolIntArray | int id** : Contient le(s) identifiant(s) de la ou des scène(s) ou du niveau à ciblé(es).
  » **float delay** : Quel est le temps mort avant le(s) destruction(s) ?

+ **Dictionary get_level_data (json = false) :** Renvoie toutes les données concernant les niveaux du jeu.
  » **bool json** : Voulez-vous renvoyer les données au format json ?

+ **bool is_defined (id) :** Est-ce que la ou les scène(s) donnée(s) sont t-elles présentes dans le niveau actuellement chargé ?
  » **String | Array | PoolStringArray | PoolIntArray | int id** : Contient le(s) identifiant(s) de la ou des scène(s) à ciblée(s).

+ **void open (id) :** Ouvre une scène ou un niveau qui a déjà été chargé(e) en mémoire. Utilisez la nomenclature précisée au niveau de la méthode *loads ()*.
  » **String | int id** : Contient l'identifiant de la scène ou du niveau à ouvrir.

IV – Les événements disponibles

+ **level_loading (data) :** Signal déclenché lorsqu'un niveau est en cours de chargement. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **String id** : Contient l'identifiant du niveau en cours de chargement.
  » **int progress** : Contient la valeur de progression du niveau en cours de chargement.

+ **level_loaded (data) :** Signal déclenché lorsqu'un niveau à finit d'être chargé. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **String id** : Contient l'identifiant du niveau ayant été chargé.
  » **int progress** : Contient la valeur de progression du niveau ayant été chargé.

+ **level_opened (id) :** Signal déclenché lorsqu'un niveau à été importé dans l'arbre des scènes.
  » **String id** : Contient l'identifiant du niveau récement ouvert.

+ **level_closed (id) :** Signal déclenché lorsqu'un niveau à été détruit ou rendu invisible.
  » **String id** : Contient l'identifiant du niveau récement détruit ou masqué.

+ **scene_loading (data) :** Signal déclenché lorsqu'une scène est en cours de chargement. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **String id** : Contient l'identifiant de la scène en cours de chargement.
  » **int progress** : Contient la valeur de progression de la scène en cours de chargement.

+ **scene_loaded (data) :** Signal déclenché lorsqu'une scène à finit d'être chargé. Cet événement renvoie un dictionaire contenant les clés suivantes :
  » **String id** : Contient l'identifiant de la scène ayant été chargée.
  » **int progress** : Contient la valeur de progression de la scène récement chargée.

+ **scene_opened (id) :** Signal déclenché lorsqu'une scène à été importé dans l'arbre des scènes.
  » **String id** : Contient l'identifiant de la scène récement ouverte.

+ **scene_closed (id) :** Signal déclenché lorsqu'une scène à été détruit ou rendu invisible.
  » **String id** : Contient l'identifiant de la scène récement détruite ou masquée.

+ **after_destroy () :** Signal déclenché après la destruction d'une scène ou d'un niveau du jeu.