

AAASeed

Version 0.69

Un logiciel « évolutionnaire »

AAASeed c'est quoi

AAASeed est un logiciel de création de monde 3D temps-réel interactif, un outil de développement, un « Lego » virtuel, un générateur d'image... AAASeed signifie une graine (Seed) de qualité 3A, c'est à dire une graine de très bonne qualité (voir les dénominations d'andouillette).

AAASeed approche technique de base

AAASeed est portable, actuellement développé et exploité sur des PCs (Windows 2000, Windows NT) sous la forme d'un fichier exécutable (.exe) entouré d'un certain nombres de dll (Dynamic Librairies).

Développé depuis 1996 entièrement en C et C++, le logiciel est indépendant du système d'exploitation, pour se faire AAASeed repose sur plusieurs composants :

- Une version légèrement étendue de la GLUT (librairie aux sources publiques) prend en charge l'interface de base avec l'OS (l'ouverture des fenêtres graphiques, la gestion des événements de base, de la souris, du clavier, des menus...
- La librairie graphique OpenGL centralise toutes les fonctions d'affichage, mais elle est déjà partiellement isolée d'AAASeed au travers d'une couche logicielle intermédiaire la GOL (voir fichiers sources gol.h et .cpp)
- Les librairies standard C et C++ et les modèles de la stl (Standart Template Library) tels que décrit dans les standards ANSI.
- Des modules spécialisés dans des taches particulières (par exemple la gestion du son) attaqués au travers d'interfaces génériques (partie intégrante de AAASeed) mais spécialisés par cibles (OS) à l'aide d'options de compilation (préprocesseur).

Les fichiers sources utilisent systématiquement le fichier ourtypes.h qui définit les types de variable en fonction de l'OS cible.

La métaphore du Synthétiseur analogique

La logique conceptuelle du logiciel AAASeed est basée sur la métaphore des synthétiseurs analogiques datant d'avant l'arrivée du numérique.

Ces derniers se présentent sous la forme de blocs physiques indépendants encapsulant chacun une fonctionnalité spécifique : clavier, oscillateur, filtre, modulateur en anneau, réverbération... Le musicien connecte alors ces différents blocs à l'aide de câbles standards, créant ainsi des processus plus complexes: deux oscillateurs dont la sortie de l'un est connectée à l'entrée fréquence de l'autre constituent un bloc de modulation de fréquence.

Les blocs d'AAASeed couvrent un champ d'application plus large, les blocs existants gèrent des primitives géométriques 2D et 3D, des primitives mathématiques, du son, de la décompression vidéo, une large gamme de capteurs d'entrée, des protocoles de communications... On peut qualifier AAASeed de « Lego virtuel », d'éditeur de monde ou bien d'environnement de développement. L'utilisation de blocs AAASeed interconnectés de manière appropriée permet de construire virtuellement toutes applications informatiques. Le champ d'application n'est limité que par les fonctionnalités des blocs existants, la puissance des machines hôtes.

Le bloc AAASeed

Cette notion de bloc (et à plus long terme de cellule) est matérialisée principalement dans les fichiers sources `obj_ui` (.h et .cpp) décrivant les classes `c_obj` et `c_obj_ui`. La quasi-totalité des classes utilisées dans AAASeed héritent de cette classe `c_obj_ui` qui prend en charge les bases de la lecture/écriture des objets (fonctions `load/save`), de l'accès aux paramètres des objets (fonctions `param`), de la relation aux autres objets (fonctions `child`, `parent`, `branch`).

De plus cette classe décrit l'interface de deux fonctions virtuelles (au sens C++) essentielles au fonctionnement d'AAASeed :

- `update()` met à jour l'objet et prépare ainsi l'appel éventuel et optimum de la fonction `draw()`.
- `draw()` «dessine» au sens large. S'il s'agit d'un objet géométrique (un cube par exemple) cette fonction prend en charge l'appel des primitives affichant l'objet en fonction des attributs courants (lumières, couleur, projection de textures, matériaux...), dans le cas d'un objet sonore cette fonction prend en charge le fait que le son associé soit entendu...

Chaque classe fille (au sens de l'héritage) de `c_obj_ui` décrit ses paramètres accessibles (`param`) ou plutôt en jargon informatique décrit son interface extérieure (c'est à dire les modalités par lesquelles on peut interagir avec lui) par l'intermédiaire d'une liste de structures de type `ST_PARAM` (fichiers sources `param.h` et `param.cpp`). La fonction `param_init_pt()` fournie par chaque classe fille de `c_obj_ui` met à jour cette description valable pour une classe entière en y rajoutant les informations spécifiques à l'instance de cet objet. Ce principe sera raffiné et étendu.

Les connections AAASeed

Si les fichiers sources connex (.h et .cpp) définissent la classe c_connex qui encapsule une connexion entre deux paramètres (structure ST_PARAM) de deux objets (classe c_obj_ui), le concept de cable standard évoqué dans la métaphore du synthétiseur se retrouve dans les fichiers sources trax (.h et .cpp) décrivant la classe c_trax.

Cette classe c_trax permet non seulement de connecter des paramètres (y compris de types différents) et de propager leurs changements, elle permet aussi d'y connecter des signaux périodiques ou aléatoires (bruit blanc, bruit gaussien, bruit de Perlin) voir des capteurs extérieurs (convertisseurs analogiques/numériques, capteurs de mouvement, d'envoyer ou des recevoir les paramètres au travers de divers liens de communications, d'enregistrer, de rejouer et de filtrer les variations temporelles de ces mêmes paramètres, d'appliquer diverses opérations mathématiques (addition, multiplication, division, inversion, moyenne...).

Cette classe c_trax est elle aussi une classe fille de c_obj_ui, ainsi tous ses paramètres sont eux aussi accessibles et connectables.

Une classe traxs décrite dans les fichiers sources traxs (.h et .cpp) englobe dans un objet c_traxs un ensemble d'objet c_trax permettant ainsi de les gérer en groupe.

En bas la classe `c_bdd`

De part sa nature au départ graphique, la structure d'AAASeed est organisé autour d'objet fils de la classe `c_bdd` (fichier source `bdd.h` et `.cpp`) qui encapsule chacun une primitive graphique, entre autres :

- `c_bdd_algo_maa`, algorithme fractal inventé par l'auteur
- `c_bdd_boid`, nuage de points avec comportement (banc d'oiseau de poisson...)
- `c_bdd_boxes`, série de boîtes rectangulaires
- `c_bdd_circle`, cercle
- `c_bdd_cone`, cône
- `c_bdd_face`, ligne, plan et volume de primitive organisée en grille
- `c_bdd_gene`, primitives géométriques tel Cube, Tétraèdre, dodécaèdre...
- `c_bdd_grid`, grille plate
- `c_bdd_mocap`, trajectoires d'animation et de capture de mouvement
- `c_bdd_null`, marqueur
- `c_bdd_movie`, affichage d'une fenêtre vidéo au premier plan
- `c_bdd_part`, système de particule très complet
- `c_bdd_proj_cone`, visualisation d'un cône de projection
- `c_bdd_quak`, niveau du jeu vidéo Quake
- `c_bdd_snd_wave`, lecture de fichier son
- `c_bdd_sound`, visualisation d'une transformé de Fourier rapide (FFT)
- `c_bdd_sphere`, sphère
- `c_bdd_torus`, tore
- `c_bdd_tri`, objet polygonal provenant de fichier `.obj` (Alias/Wavefront) ou `.geo`
- `c_bdd_tube_path`, tube extrudé le long d'une ligne tridimensionnelle
- `c_bddtex2d`, Texte typographique (Police TrueType)
- ...

Certains de ces objets descendent de classe intermédiaires :

`c_bdd_multiple` (fichier `bdd.h` et `.cpp`) qui redéfinit les modalités d'affichage de chaque point. Par exemple une sphère peut être dessinée à l'aide de cube attaché à chacun de ses points et orienté suivant ses normales. Cette classe, permet aussi de gérer des appels en boucles itératives à d'autres structures de plus haut niveaux (`c_layer` et `c_layers` décrits ci-dessous).

`c_bdd_uv` (fichier `bdd_uv.h` et `.cpp`) regroupe toutes les fonctionnalités d'objet assimilable à une grille bidimensionnelle déformée (`c_bdd_grid`, `c_bdd_sphere`, `c_bdd_torus`...).

Au dessus la classe c_layer

Les objets c_bdd sont contenus dans des couches définies sous forme d'objet c_layer (fichiers sources layer.h et .cpp) qui se chargent de préparer les conditions du rendu optimum en fonctions des attributs courants. L'objet c_layer contient éventuellement des instances des objets suivant :

- c_def_node: déforme point par point la géométrie des objets
- c_render: attributs de rendu 3d
- c_multiple : attribut de rendu multiple concerne les objets c_bdd_multiple
- c_map: attribut de plaquage de texture
- c_color: attribut de couleur
- c_lights_switch: commutateurs de lumière
- c_tex_anim: animation de texture
- c_feedback: processus de feedback image (dis parfois autoMap)
- c_model: information de taille et de position
- c_tex_video: lecture de séquence vidéo dans les textures
- c_transfo: rotation, échelle, translation... dites transformations
- c_fog: brouillard
- c_material: matériaux (diffus, spéculaire...)

l'objet c_layer gère directement au travers de paramètres la commutation et l'utilisation des objets ci-dessus, ainsi que d'autres options. Pour toutes ces classes existe un pointeur sur l'objet courant (variables *_cur) que chaque objet peu donc utiliser ou changer. La classe c_layer contient des commutateurs pour chaque classe ci-dessus, permettant d'utiliser l'objet courant, de lui substituer son propre objet voir de le désactiver. Ainsi, exemple simple, un objet couleur peu contrôler le rendu d'une série de couches successives (au sens de l'ordre de rendu).

Puis la classe c_layers

La classe c_layers (fichiers sources layers.h et .cpp) est un groupe de 26 couches (objets c_layer de A à Z) dont elle gère l'affichage successif ainsi qu'éventuellement d'autres options et objets :

- c_obj_value: une série de paramètres neutres disponibles comme variables
- c_seedcam : un objet caméra définissant le point de vue
- c_traxs: pistes dites locales connectables dans l'objet et ses sous-objets
- c_transfo: transformation appliqué à toutes les couches internes

Encore plus global avec la classe c_app

Cette classe application (c_app dans les fichiers sources app.h et .cpp) gère un ensemble de groupes (ensemble d'objet c_layers) et contrôle des objets et attributs associés à l'ensemble des groupes. Cet objet est pour l'instant instancié une seule fois pour des raisons historiques.

En particulier c_app contient un objet c_traxs qualifié de pistes globales.

Et enfin la boucle de « rendu »

La librairie Glut agit comme boucle d'événements et appelle AAASeed (fonctions de callback). Une de ces fonctions de callback appelle la fonction render() (fichier seeddraw.cpp) qui traite la majorité des tâches.

De plus à ce niveau on trouve des structures de banques (d'image, de matériaux par exemple) que les classes ci-dessus peuvent sélectionner juste par un numéro d'index, voir par un numéro de banque.

On peut résumer son déroulement de la manière suivante :

- Mise à jour des objets globaux
 - temps
 - interface
 - capteur
 - communication
 - attributs
 - banques
 - ... voir fichier source
- Mise à jour de app (objet c_app)
 - C_APP**
 - Mise à jour des pistes globales (**C_TRAXS**)
 - Appel de draw() pour tout les c_layers actifs
 - C_LAYERS**
 - Mise à jour des traxs locales (**C_TRAXS**)
 - Mise à jour éventuelle de la camera et des transformations
 - Appel de update() puis draw() pour tout les c_layer actifs
 - C_LAYER**
 - Mise à jour de tout les objets nécessaires au rendu
 - Appel éventuel de update() puis draw() de l'objet c_bdd actif
 - C_BDD**
 - Appel éventuel d'objet **C_BDD**, **C_LAYER** ou **C_LAYERS**
 - En cas de stéréoscopie nouvel appel de draw() pour tout les c_layers actifs
 - C_LAYERS...**
 - En cas de double click nouvel appel de draw() pour trouver l'objet cliqué
 - C_LAYERS...**
 - Dessin éventuel de l'interface
 - Attente éventuelle du retour de trame
 - Sauvegarde éventuelle de l'image rendue

En jargon informatique on peut parler d'arbre statique de rendu de scène. Mais si pour l'instant la structure est prédéfinie dans son organisation, les mécanismes de banques, de pistes (traxs), de commutations, d'objets courants, d'appel de couches et de groupes à bas niveau de l'arbre (c_bdd_multiple) donnent au logiciel une extrême flexibilité et une grande richesse.

L'interface actuelle

L'interface visualise sous forme textuelle l'arborescence, chaque niveau étant indenté, on peut ouvrir/fermer chaque branche de l'arbre. Les pistes (traxs) sont représentés par de nouvelles branches là où elles sont connectées. De nombreux raccourcis clavier permettent de naviguer et de modifier les paramètres. Il suffit de cliquer avec le bouton de droite de la souris et de la déplacer pour modifier un paramètre.

L'effet est immédiat puisque la boucle de rendu est appelée en général de 120 à 25 par seconde, y compris lorsque les menus sont utilisés ou que des boîtes de dialogue traditionnelles sont utilisées.

Cet aspect temps réel est une caractéristique majeure du logiciel et un avantage réel. Il n'y a pas de distinction entre « l'édition » et « l'utilisation ».

Le bouton de gauche agit sur les paramètres comme un menu contextuel qui permet en particulier de connecter et de déconnecter les pistes.

Le bouton de droite déroule le menu général.

Optimisations

L'utilisation intensive des templates, des "inline", de la stl, des déroulement de boucles, des mémorisation d'état, de minimisation des appels de fonctions à bas niveaux ajouté à une structure adaptée et contrôlable par l'utilisateur, font de AAASeed un programme performant en terme de vitesse.

Dynamique et sensibilité

La précision et de sensibilité (dynamique la plus large possible) est une préoccupation à toutes les étapes. Ainsi, sauf raisons contraires, un paramètre est par défaut un nombre réel dans AAASeed. Par exemple les couleurs ne sont pas des nombres entiers entre 0 et 255 mais des réels entre 0. et 1. .

Allié à l'aspect temps-réel du logiciel, la conséquence est une réactivité, une sensibilité de AAASeed qualifiée de « biologique » par certains utilisateurs. De plus, le logiciel absorber ainsi facilement la montée en puissance permanente du matériel.

Un logiciel « évolutionnaire »

AAASeed a été développé de manière incrémentale, et contraint en permanence à de nombreuses et diverses logiques de productions, en particulier sur des scènes ou dans des installations publiques. Ainsi il s'est « adapté », à été façonné par son environnement.

Cet aspect est difficile à expliquer globalement, mais se retrouve par exemple dans des choix de structures, d'algorithmes, de noms, de regroupement de variables... Malgré son interface encore primaire AAASeed a été « poli », « patiné » et les créateurs qui l'utilisent sont sensibles à cette dimension. Autrement dit, il s'agit plus d'un produit d'artisan, de compagnon, que d'une froide production logique.