

Licence d'utilisation et d'exploitation :

Copyright (c) 2004, iri

Permission vous est donnée de copier, distribuer et/ou modifier ce document selon les termes de la Licence GNU Free Documentation licence (FDL), Version 1.1 ou ultérieure publiée par la Free Software Foundation;
avec les sections inaltérables suivantes :

Infos générales initiales

But

avec le texte de première page de couverture suivant :

aucun texte de première page de couverture

avec le texte de dernière page de couverture suivant :

aucun texte de dernière page de couverture

Note : Une traduction en français de cette Licence est disponible ici :

<http://cesarx.free.fr/gfdlf.html>.

L'original de cette licence est disponible ici : <http://www.gnu.org/licenses/fdl.html>.

Historique :

version 1.1 : septembre 2004

- ajout de définitions de mots-clés
- ajout de paragraphes et de notes
- modification du contenu de certains pararaphes,
- modification de certains points de présentation.

version 1.0 : octobre 2002

- première version publique

Infos générales initiales :

- auteur : iri
- email : info@irizone.net
- web : <http://www.irizone.net> (iriZone 2D)
- licence : FDL
- support : par mail ou via le forum d'iriZone 2d
- date de première mise en ligne publique : octobre 2002
- l'auteur ne saurait être tenu responsable de tout dommage matériel ou immatériel suite au téléchargement ou à l'utilisation de ce document qui est fourni ad hoc.

But :

Création d'un site scol contenant, outre une scène 3D et les éléments déjà intégrés dans les tutoriaux de niveau 1 & 2 :

- des plugins 3d (instances, ancrs et liens)
- des boutons interactifs
- un tchat vocal
- la diffusion aléatoire d'images, de sons, ... hors de la 3d à intervalle régulier
- un tableau blanc (avec condition d'utilisation)
- l'affichage d'une liste d'action possible
- une bannière d'affichage de news
- un bot (robot).

Cela se traduira par l'utilisation des modules suivants :

- plugins du C3D3,
- modules graphicsCheckBox, bot0, whiteboard, banner, listBox, mediaSequence, timer, speaker et random

L'objectif final de cette série de tutoriels est de rendre le futur scolmaster (créateur de monde Scol) suffisamment autonome pour comprendre la logique d'intégration et de comprendre les différentes documentations relatives à des projets plus poussés.

Ce troisième tutoriel vous donnera les bases suffisantes, je l'espère en tout cas !, de votre prochaine autonomie avec le SCS ! :)

Pré-requis :

- un SCS version 2 ou supérieure correctement installé, avec toutes les librairies (modules et graphiques) par défaut.
- connaître les bases du fonctionnement du SCS (cf autres tutoriaux ou aide), notamment l'aide du SCS.
- Connaître et comprendre le tutoriel "Création d'un site scol - Niveau 2" du même auteur, disponible sur son site. Les manipulations déjà expliquées ne seront pas reprises ici. Ce tutoriel sera donc un peu plus complexe que les précédents.

Informations :

Avec ce tutorial, deux fichiers et un dossier étaient joints (fichiers sources) :

- demolri_3.scol
- demolri_3.dms

Shématiquement, le premier est le fichier script lançant le serveur du site.

Le second contient toutes les infos de conception du site.

Un dossier "img" concernant trois fichiers : sitemapON.jpg , sitemapOFF.jpg et demolributton.jpg

Un dossier "media" contenant six fichiers : Coffee Bean.bmp , flourish.mid , Gone Fishing.bmp , onestop.mid , Rhododendron.bmp et town.mid

Créez (s'il n'existe pas) le dossier "demolri" dans ..\scol\partition\worlds et placez ces deux dossiers et ces deux fichiers dans le dossier "demolri".

(..\scol\partition\worlds\demolri\).

Ecrasez le dossier "img" s'il en existe déjà un.

Le but n'étant pas de créer une cellule 3D, le module C3D3 ne sera pas détaillé ici !

Mise en oeuvre :

Plusieurs solutions sont possibles. En voilà une :

après copie des deux fichiers aux bons emplacements, double-cliquez sur demolri_3.scol pour lancer le serveur.

Une fois ouvert, cliquez sur le bouton "Connection" pour une session en local (sans passer par le Net). Vous pourrez voir l'aspect général du site.

Ensuite, ouvrez le SCS puis le fichier demolri_3.dms afin d'avoir un modèle puis ouvrez demolri_2.dms dans lequel vous ajouterez ces nouvelles fonctions.

Profitez-en pour ouvrir l'Arbre de création (touche F6 pour le raccourci clavier).

Vous allez recréer le site à l'identique (la comparaison entre les deux sites permettra de mieux repérer les causes d'éventuels dysfonctionnements).

Note : demolri_3 commence exactement à la fin de demolri_2. De ce fait, demolri_2 est inclus dans demolri_3.

A/ Plugins 3d :

Ce sont des modules spécifiques s'interfaçant directement avec le C3D3. Ils s'utilisent et se configurent donc dans le module 3d. Tous, sans exception, doivent être placés dans le dossier plugins du C3D3 pour être reconnus par ce dernier.

Les cellules 3d ont déjà été créées dans les deux tutoriaux précédents (niveau 1 et 2). Je n'y reviens donc pas.

A1/ Target et généralités sur les instances :

« Target » permet de toujours orienter un ou plusieurs objets vers le visiteur.

Dans la "salle 1", placez la petite fille près du bureau déjà en place : (lib/3d/furnitur/cartoon/toy/toy0007.m3d). Associez-lui le lien "fille".

Pour qu'elle se tourne en permanence vers le visiteur, nous allons lui associer ce plug in.

Schématiquement, une instance est une "copie" d'un plugin et possédant certains paramètres (pouvant être différents suivant la copie). Dans la plupart des cas, il est ainsi possible de faire cohabiter plusieurs instances d'un même plugin avec des configurations différentes, le tout dans une même scène 3d (mais sur des objets distincts).

En règle générale, si plusieurs matériaux et/ou textures et/ou objets utilisent le même plugin avec la même configuration, créez une ancre contenant ces objets et créez ainsi une seule instance liée à cette ancre.

De base, à toute instance sont définis un nom, le plugin dont elle est la « copie » (la « classe ») et les objets sur lesquels elle agira (l' « ancre »).

Eventuellement, selon le type de plugin, une configuration spécifique peut-être nécessaire.

Dans le C3D3, cliquez sur le menu déroulant Objets pour afficher Instances. Vous devriez avoir déjà l'instance "inst_snapAvatar(SnapAvatar) <>" présente.

Cliquez sur Ajouter :

- Dans le champ Nom, tapez : « inst_target »
- Dans le menu Classe, choisissez « Target(3.1) »
- Dans le menu Ancre, sélectionnez « fille »
- Enfin, dans paramètre, cliquez devant *Verticale target*.

Cliquez sur OK pour Valider.

Verticale Target n'orientera la petite fille uniquement suivant l'axe verticale : elle se tournera vers sa gauche ou sa droite suivant les déplacements du visiteur.

Lorsqu'il n'y a qu'un seul objet affecté par une instance, il est inutile de créer une ancre, il est possible, comme ici, de lui affecter directement le lien associé à l'objet. Si ni lien ni ancre ne sont associés à une instance, alors celle-ci restera sans effet (sauf cas particulier des plugins agissant sur la scène 3d entière).

Note : certains plugins agissent indépendamment sur chaque visiteur (client). C'est le cas de Target : la petite fille sera toujours face au visiteur. En conséquence, elle n'a pas d'orientation "absolue". A l'inverse, d'autres ont le même comportement chez tous les visiteurs (tels shift ou rotate), d'autres peuvent combiner les deux, au choix du concepteur, comme realVideo.

Enfin, certains plugin peuvent avoir des actions différentes ou générer des événements ce qui permet de les rendre interactifs. Des liens vers d'autres modules ou vers eux-même pourront être établis dans le SCS. Target ne possède pas ce privilège.

Une fois le site en ligne, dès qu'un lien est associé à un objet, le curseur de la souris se transformera en main lorsque celui-ci sera survolé indiquant qu'il pourra être cliqué. Pour l'éviter, choisissez l'onglet « Liens » et sélectionnez « fille ». Cliquez alors sur le bouton Invisible (Visible).

Validez l'ensemble en cliquant sur le bouton Ok du C3D3.

Note : On emploie souvent le mot « client ». Il ne s'agit pas d'une approche commerciale mais de l'opposition entre un « serveur » et ses « clients ». Le client peut être assimilé au « visiteur ».

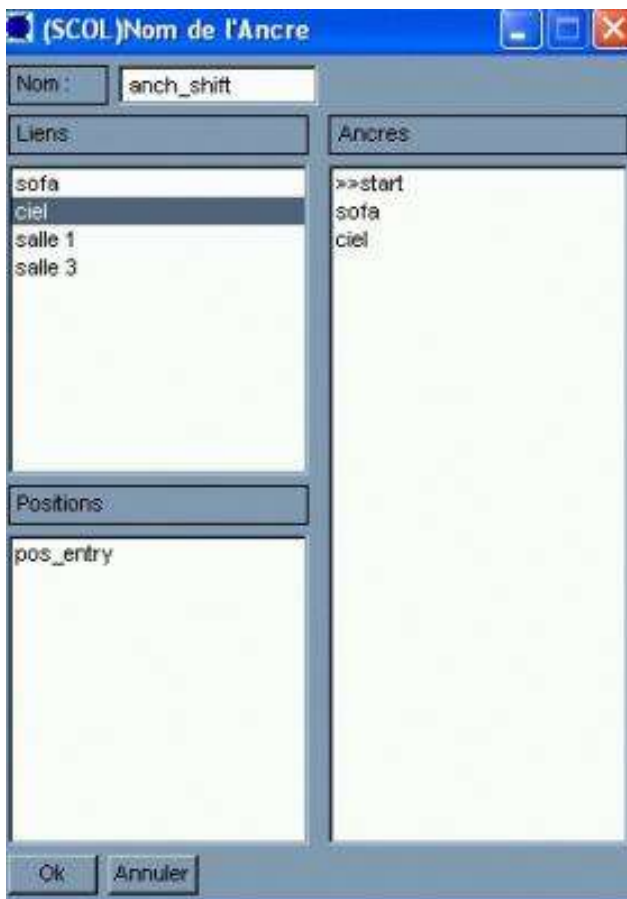
A2 : Shift:

Il permet le glissement de la texture suivant les valeurs données dans l'éditeur de l'instance.

Editez le C3D3 correspondant à la "salle 2". Sur le plafond, mettez un lien "ciel" invisible. Ajoutez un sofa (lib/3d/furnitur/cartoon/sofa/sof0002.m3d) et associez un lien invisible "sofa" au matériau MAT01-31 (la texture rouge).

Sur ces deux matériaux, vous allez appliquer le plugin Shift.

Remarquez d'abord que, contrairement à Target, c'est sur un matériau particulier que le plugin va agir et non sur l'objet entier. De plus, vous devez créer une ancre pour associer ces deux matériaux à une même instance.



Dans le menu déroulant Objets, cliquez sur Ancres (Anchors en anglais) puis sur le bouton Ajouter.

- Dans le champ Nom, saisissez "anch_shift"
- Dans la liste Liens, double-cliquez sur "sofa" puis sur "ciel" afin de les faire apparaître dans le champ Ancres (à droite).

Cliquez sur Ok pour valider.

Certains plugins (comme Trajectory) demandent des positions. Double-cliquez sur les positions voulues pour les faire apparaître à droite (pour Shift, ce n'est pas la peine !). Selon l'effet recherché, vous pouvez prendre des positions plusieurs fois.

Rappel : Pour supprimer un lien ou une position déjà à droite, double-cliquez dessus pour l'enlever.

Ainsi, dans le champ Ancres sont affichés tous les éléments qui seront utilisés dans l'instance à laquelle cette ancre sera liée. C'est pourquoi, entre autres, il convient de lui donner un nom explicite !

Il n'est pas possible d'affecter plusieurs ancres à une même instance, en revanche, vous pouvez associer une même ancre à plusieurs instances distinctes. Dans ce cas, les liens et/ou positions contenus dans l'ancre seront utilisés par les instances qui y sont liées.

Dans le menu déroulant *Ancres*, choisissez *Instances*. Cliquez sur *Ajouter* :

- Nommez-la "inst_shift",
- affectez-lui la classe "Shift 2.2"
- et l'ancre "anch_shift".
- Entrez les valeurs que vous voulez dans le champ Paramètres. U et V donnent le décalage des coordonnées de la texture. Plus ils seront grands, plus le décalage sera rapide. Ces deux valeurs peuvent ne pas être identiques.
- Cliquez sur *Ok* pour valider.

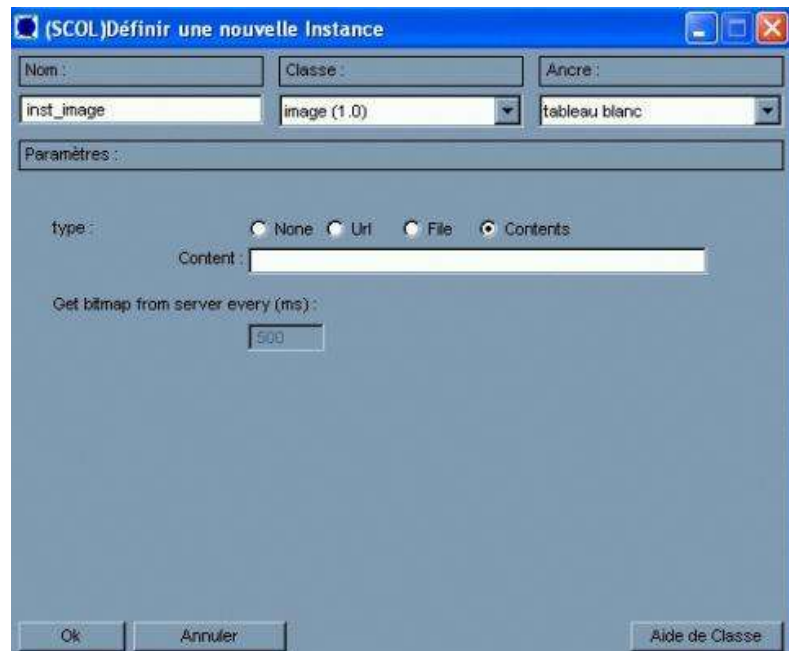
A3 : Image :

Il permet d'afficher sur un matériau des images provenant de différentes sources (serveur, net, etc ..). Il permet aussi l'affichage du tableau blanc. C'est pour cette dernière raison que vous l'utilisez ici.

Sur le mur en face du sofa, placez un lien invisible "tableau blanc". Ajouter une fleur à côté du pan de mur précédent (lib/3d/furnitur/cartoon/diverse/div0009.m3d). Associez-lui le lien (visible cette fois !) "Dessin".

Allez à l'onglet *Instances* pour en créer une nouvelle que vous appellerez "inst_image". Vous lui associez la classe "image 1.0" (en bas de liste, Scol fait la différence maj/min) et le lien "tableau blanc" pour l'ancre.

Dans *Paramètres*, sélectionnez "Contents". Laissez vide le champ « Content ». Validez par *Ok*.



Note : *Url* permet d'afficher une image présente sur le net tandis que *File* la cherchera dans le dossier *Partition* du serveur.

Dans ces deux cas, vous pouvez spécifier l'adresse ou le chemin relatif du fichier image en question dans le champ "Content". Autrement, vous l'indiquerez dans les paramètres du lien sous le SCS. En effet, le plugin *Image* permet une interactivité plus forte en générant événements et acceptant certaines actions.

Get bitmap from server every ms donne la cadence de rafraichissement de l'image (dans les cas *Url* et *File* uniquement).

D'autres plugins ont déjà été intégrés dans le tutorial précédent (niveau 2). Fermez le C3D3 en appuyant sur *Ok*.

B/ Module Whiteboard et conditions :

Il permet l'affichage et la gestion d'un tableau blanc : l'utilisateur peut dessiner grâce à divers outils présent sur une interface spécifique et le dessin s'affiche au fur et à mesure chez les autres visiteurs. Plusieurs utilisateurs peuvent s'y connecter simultanément.

Pour l'exemple, vous restreindrez l'accès à un seul visiteur à la fois qui prendra le pseudo "dessinateur". Cela permettra de voir différentes actions.

Ouvrez l'arbre de création du SCS (touche F6), allez dans "commTools" et double-cliquez sur Whiteboard. Validez par Ok.

Placez les liens suivants :

salle 2.in >> whiteboard.start

Charge au démarrage du site le module du tableau blanc

salle 2.dessin >> whiteboard.show

Affiche l'interface de dessin au visiteur.

Editez ce lien (en double-cliquant dessus dans le SCS) et sélectionnez-le dans la liste.

Ajoutez la *condition* (dans le champ « condition sur le lien ») : "login dessinateur" , sans les guillemets comme illustré ci-dessous.

Ainsi, le visiteur devra prendre le pseudo "dessinateur" pour avoir le tableau blanc et devra le céder pour qu'un autre puisse le prendre.



login.loginChanged >> whiteboard.hide

selon le pseudo du visiteur, l'interface de dessin disparaîtra.

Entrez la condition de la même façon que précédemment : "notlogin dessinateur" (sans les guillemets). Le "not" inverse la condition. Ici, tous les visiteurs avec un pseudo différent de "dessinateur" verront le tableau s'effacer (par exemple si l'utilisateur change de pseudo entre temps). Notez que ça n'affecte en rien l'affichage du résultat sur le mur !

Vous pouvez modifier vos bannières une fois connecté au site et les enregistrer. Pour avoir cette possibilité, veuillez à cocher la case, en bas de l'éditeur, « sauver les modifications effectuées en dynamiques ».

Pour éviter que n'importe quel visiteur modifie vos bannières, nous allons limiter cette fonction à l'interface d'admin :

- fermez l'éditeur « banner ».
- éditez le module Control 3.1 déjà intégré
- rajoutez "Bannière".
- Fermez-le.

Créez les liens suivants :

shell.start >> banner.start

Charge en mémoire ce module au lancement du site.

banner.bannerClick >> inOut.openUrl

lorsque le visiteur clique sur une bannière reliée à une url, celle-ci sera lancée

Eventuellement,

control.Banniere >> banner.edit

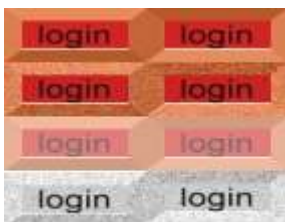
si vous voulez éditer en ligne vos bannières (cf ci-dessus).

Dans le module Client du SCS (F3), placez une zone sous celle de la 3d (en diminuant un peu cette dernière). Vous affecterez à cette nouvelle zone le module Banner.banner.

D/ Module graphicCheckBox

Il permet de personnaliser les boutons de l'interface. Chaque bouton peut contenir ses différents états (normal, normal survolé, cliqué, cliqué survolé, inactif et le masque (zone sensible au rollover).

Note : « survolé » ou « rollover » indique « survolé par la souris du visiteur ».



D1 : En premier lieu, le bouton doit être pensé sur le papier. Vous avez à gauche un superbe bouton qui va répondre aux plus grandes exigences de ce module ! Il servira d'exemple car tous les fichiers images des boutons auront la même structure.

Chaque élément possède la taille du bouton, tel qu'il sera intégré sur l'interface.

La colonne de gauche concerne les états non cochés (non appuyés par la souris du visiteur), celle de droite, les états cochés.

- La ligne 1 correspond au bouton non survolé par la souris
- La ligne 2 correspond au bouton survolé (rollover)
- La ligne 3 correspond au bouton cliqué
- La ligne 4 correspond à l'état inactif (optionel, si oui, il est impératif de cocher la case en bas de l'éditeur)
- La ligne 5 correspond au masque du bouton : en noir les zones non sensibles au survol de la souris, en blanc, les zones sensibles (ici, toute la surface du bouton est sensible dans les deux cas, donc blanche, même si cela ne se voit pas avec le fond blanc de la page !)

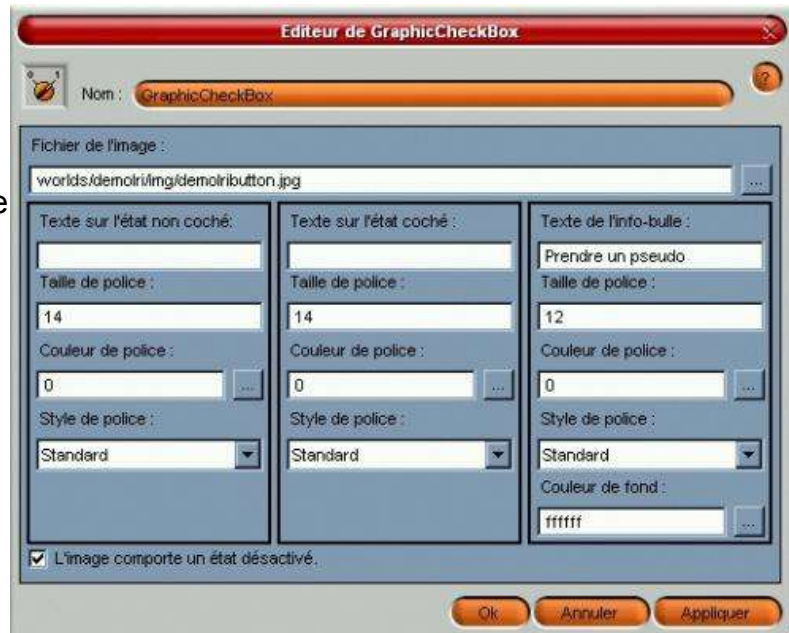
Enregistrez-le au format jpg, png ou bmp dans un sous dossier de .../scol/partition.

D2 : Vous trouverez ce module dans Interf de l'arbre de création (F6 !).

Sélectionnez le fichier image du bouton en appuyant sur le bouton [...]. Celui-ci doit obligatoirement se trouver dans le dossier Partition. Remplissez l'éditeur comme vous le souhaitez. Les textes affichés peuvent être différents dans les trois cas et même être vides si le texte est déjà intégré à l'image, comme c'est le cas dans cette illustration.

Si votre bouton comporte un état inactif, il faut alors cocher la case "L'image comporte un état désactivé".

Cet état désactivé peut être utile si on désire limiter l'utilisation d'un bouton sous certaines conditions. Par exemple, mettre un bouton "Whiteboard" mais le rendre actif uniquement si le visiteur se trouve dans "salle 2".



D3 : Au départ, le bouton est affiché dans l'état "ligne 1 - colonne gauche".

Il peut être modifié grâce aux actions suivantes :

module.event >> *GraphicCheckBox.show* => Affiche le bouton chez le visiteur (client)

module.event >> *GraphicCheckBox.hide* => Cache le bouton de l'interface cliente

module.event >> *GraphicCheckBox.enable* => Active le bouton (il est utilisable)

module.event >> *GraphicCheckBox.disable* => Désactive le bouton (il n'est plus utilisable)

module.event >> *GraphicCheckBox.check* => Force à l'état coché le bouton depuis un autre module. Ceci est possible même si le bouton est à l'état 'désactivé'.

module.event >> *GraphicCheckBox.uncheck* => Force à l'état décoché le bouton depuis un autre module. Ceci est possible même si le bouton est à l'état 'désactivé'.

Note : *[module].[event]* indique n'importe quel événement (event) de n'importe quel module.

Et les événements suivants :

GraphicCheckBox.enabled >> *module.action* => L'activation du bouton engendre une action sur un module

GraphicCheckBox.disable >> *module.action* => La désactivation du bouton déclenche une action sur un module

GraphicCheckBox.check >> *module.action* => Le changement d'état du bouton engendre une action sur un module

GraphicCheckBox.uncheck >> *module.action* => Le changement d'état du bouton engendre une action sur un module

Il existe les autres actions et événements plus classiques :

shell.start >> *GraphicCheckBox.start* => Permet de démarrer le module et d'afficher le bouton dans l'interface graphique cliente

module.event >> *GraphicCheckBox.destroy* => Arrête le module *GraphicCheckBox* chez le client

GraphicCheckBox.in >> *module.action* => La partie cliente du module est démarrée et prête à recevoir des événements

GraphicCheckBox.out >> *module.action* => La destruction du module engendre une action

GraphicCheckBox.entering >> *module.action* => La partie cliente du module va être démarrée et cela engendre une action

GraphicCheckBox.shown >> *module.action* => L'affichage du bouton engendre une action

GraphicCheckBox.hidden >> *module.action* => La disparition du bouton sur le client déclenche une action

D4 : Ici, vous devez créer les trois suivants :

shell.start >> *graphicCheckBox.start*

graphicCheckBox.check >> *login.show*

graphicCheckBox.check >> *graphicCheckBox.uncheck* (afin de remettre à non coché le bouton)

Pour une interactivité plus grande, n'hésitez pas à user des actions et événements disponible ! :)

Supprimez l'ancien bouton (module « bouton ») *Pseudo* ! Dans le module Client (F3), affectez à l'ancienne zone du bouton *Pseudo*, le module *graphicChexBoxLogin.button* !

Répétez la manip si vous le souhaitez (création du fichier image et module, un module par bouton) pour les autres boutons !

E/ Module Speaker :

Il permet les tchats vocaux. Une personne parle, les autres écoutent. Deux personnes ne peuvent pas parler simultanément avec ce module.

Dans l'arbre de création du SCS, allez dans Sounds et double-cliquez sur Speaker 3.0. Validez l'éditeur par Ok.

Les boutons d'écoute et d'enregistrement sont intégrés automatiquement. Si vous les changez, modifiez les bmp en place dans le dossier du module (dms/sound/speaker).

Créez le lien suivant :

shell.in >> *speaker.start_direct*

afin d'avoir une connexion directe de client à client sans surcharger la bande passante du serveur.

Dans le module Client (F3), créez trois zones :

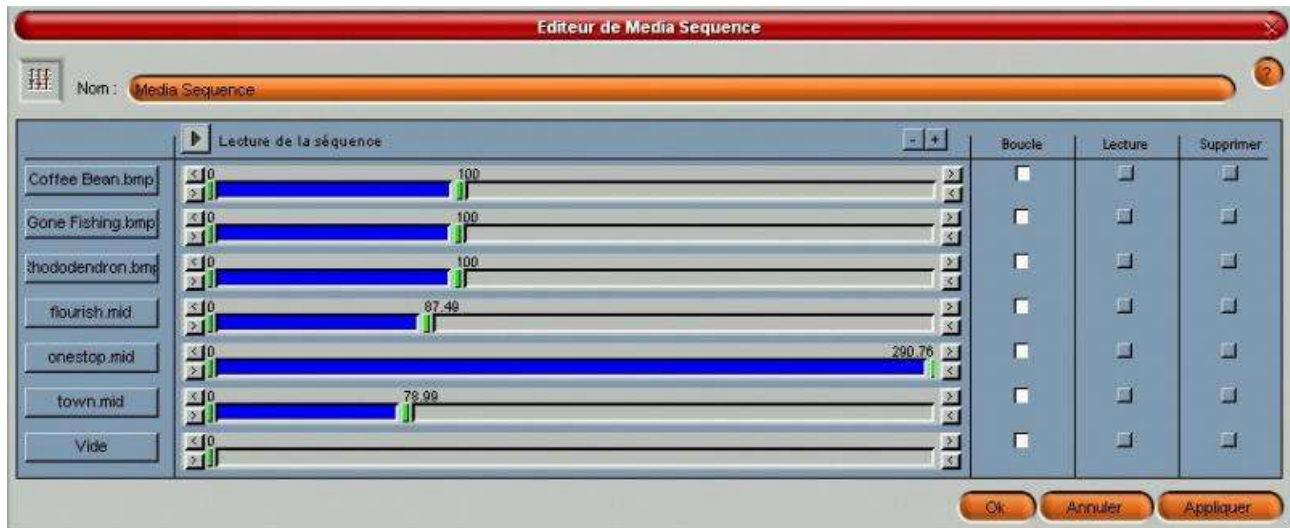
- une d'une taille de 16 par 16 pixels appelée "play" à laquelle vous associez "speaker.buttonPlay"
- une autre de même taille appelée "rec" à laquelle vous associez "speaker.buttonRec"
- une troisième d'une taille variable, nommée "qui parle" associée à "speaker.info". C'est dans cette zone que sera inscrit le pseudo de la personne en train de parler.

Note : il existe un plugin du C3D3, *soundDialog*, plus complet et plus configurable que *speaker*. L'auteur a également développé un module « *longspeaker* » offrant des possibilités supplémentaires. Il est disponible librement sur son site.

F/ Module MediaSequence :

Ce module permet la diffusion d'image jpeg ou bmp, de fichiers son wav ou midi ou de fichiers vidéo mpeg ou avi indépendamment de la 3d.

Dans ce tutorial, dès l'entrée dans la "salle 3" un son ou une image sera diffusé aléatoirement à chaque intervalle de temps prédéterminé.



Sélectionnez Media de l'arbre de création du SCS puis double-cliquez sur Media Sequence 3.0 afin d'ouvrir l'éditeur illustré ci-dessus. Pour ajouter un media, cliquez sur le bouton "Vide" (!) et sélectionnez votre fichier qui doit se trouver dans le dossier Partition du serveur. Les six fichiers présents ici se trouvent dans le sous-dossier media de demolri. Validez par Ok.

Dans le module Client (F3), créez un sous-document (document fils) en appuyant sur la touche Retour Chariot du clavier. Nommez-le "Media". A l'intérieur de ce dernier, placez deux zones, l'une de 158 par 128 pixels, appelée "image", l'autre de taille libre, appelée "midi". Affectez à la première les trois midis, à la seconde, les trois bmp.

Les liens de ce module seront créés plus loin.

G/ Module Random :

Il génère aléatoirement une sortie parmi un nombre de sortie possible déterminé à l'avance.

Vous l'utiliserez ici pour diffuser aléatoirement les fichiers que vous avez ajouté à MediaSequence.

Editez-le en double-cliquant sur Random 3.1 dans l'item Tools de l'arbre de création (F6). Puisqu'il y a six média à diffuser aléatoirement, entrez "6" dans l'éditeur et validez par la touche Ok.

Les liens seront créés plus tard.

H/ Module Timer :

Ce module génère un "top" à intervalle de temps régulier permettant ainsi d'effectuer une action périodiquement.

Dans l'item Tools de l'arbre de création, double-cliquez sur Timer 3.1. Saisissez "30000" pour avoir un top toutes les 30 secondes (la valeur à saisir est en effet en milliseconde).

Validez en cliquant sur Ok.

Créez les liens suivants pour les trois modules précédents :

salle 3.in >> timer.start

lorsque le visiteur est dans la salle 3, le chrono commence

salle 3.in >> mediaSequence.start

lorsque le visiteur est dans la salle 3, le module est chargé

timer.top >> random.input

à chaque top un demande de sortie aléatoire est envoyée

random.out1 >> mediaSequence.media1

random.out2 >> mediaSequence.media2

...

random.out6 >> mediaSequence.media6

selon la valeur renvoyé aléatoirement par le module Random, un certain media (fichier) sera diffusé.

Note : le premier media sera diffusé après 30 secondes de présence effective dans la salle. Pour avoir une première lecture immédiate, rajoutez le lien suivant :

salle 3.in >> random.input

I/ Module listBox :

Ce module affiche une liste dont les éléments peuvent non seulement être cliqués mais aussi ajoutés ou soustraits.

Dans l'arbre de création, allez à Interf et double-cliquez sur listBox 3.0. Dans l'éditeur, ajoutez par exemple des adresses web.

Cochez les cases "Même liste pour tous les clients" et "Le nom de l'item cliqué sera passé en paramètre". Validez par Ok.

Note : Si vous ne voulez pas que tous les clients aient la même liste (par exemple quand il est possible d'ajouter des éléments dans la liste via un module Input ou autres), ne cochez pas la première case. De même, vous pouvez mettre des noms plus évocateurs et passer le lien dans le paramètre du lien correspondant vers le module inOut. Il faudra alors paramétrer chacun des éléments et décochez la seconde case.

Créez les liens suivants :

shell.start >> listBox.start

charge le module à l'ouverture du site

listBox.click >> inOut.openUrl

ouvre l'url correspondante lors du clic du visiteur sur un item particulier de la liste.

Dans le module Client (F3) créez un document fils "Liste Web", placez-y une zone qui sera associée à ce module (listBox.list).

Note : la version listBox 2.1 contient un bug. Il a été corrigé avec la version 3.0. Si vous ne la possédez pas, téléchargez-la librement sur le Scolring (<http://www.scolring.org>), rubrique « Download » puis « Modules DMS ».

J/ Module Bot0 :

Il s'agit d'un bot programmable à plusieurs niveaux. Dans des cas simples, l'éditeur sera suffisant. Dans les autres configurations, vous devrez ajouter vos propres commandes Scol.

Vous trouverez ce module dans l'item Bot de l'arbre de création. Dans l'éditeur, ajoutez les mots-clés pour sa conversation puis validez par Ok.

Ici, les 4 mots

- "Bonjour",
- "Salut",
- "Ciao" et
- "Bye" sont saisis.

Dès que le visiteur saisira l'un de ces mots, le bot réagira (ici par un message dans le tchat, mais cela peut-être n'importe quelle action !).

Note : D'autres bots circulent plus ou moins officiellement, plus ou moins configurable. Celui-ci est fourni en standard avec le SCS 2 et supérieur.

Créez les liens suivants :

shell.in >> *bot.register*

shell.out >> *bot.unregister*

ces deux liens prennent en compte l'arrivée et la déconnexion d'un visiteur.

salle 1.spy >> *bot0.!hear*

salle 2.spy >> *bot0.!hear*

salle 3.spy >> *bot0.!hear*

ces trois liens permettent au bot d' « espionner » ce qui se dit dans chacune des salles pour intervenir le cas échéant.

bot0.bonjour >> *colorTerm.broadMsg* + paramètre

bot0.salut >> *colorTerm.broadMsg* + paramètre

bot0.ciao >> *colorTerm.broadMsg* + paramètre

bot0.bye >> *colorTerm.broadMsg* + paramètre

ces quatre liens donnent la réaction du bot à chaque mot clé lu sur le tchat. Le paramètre est la phrase qu'il affichera dans le tchat (module colorTerm), voyez l'image ci-dessous.



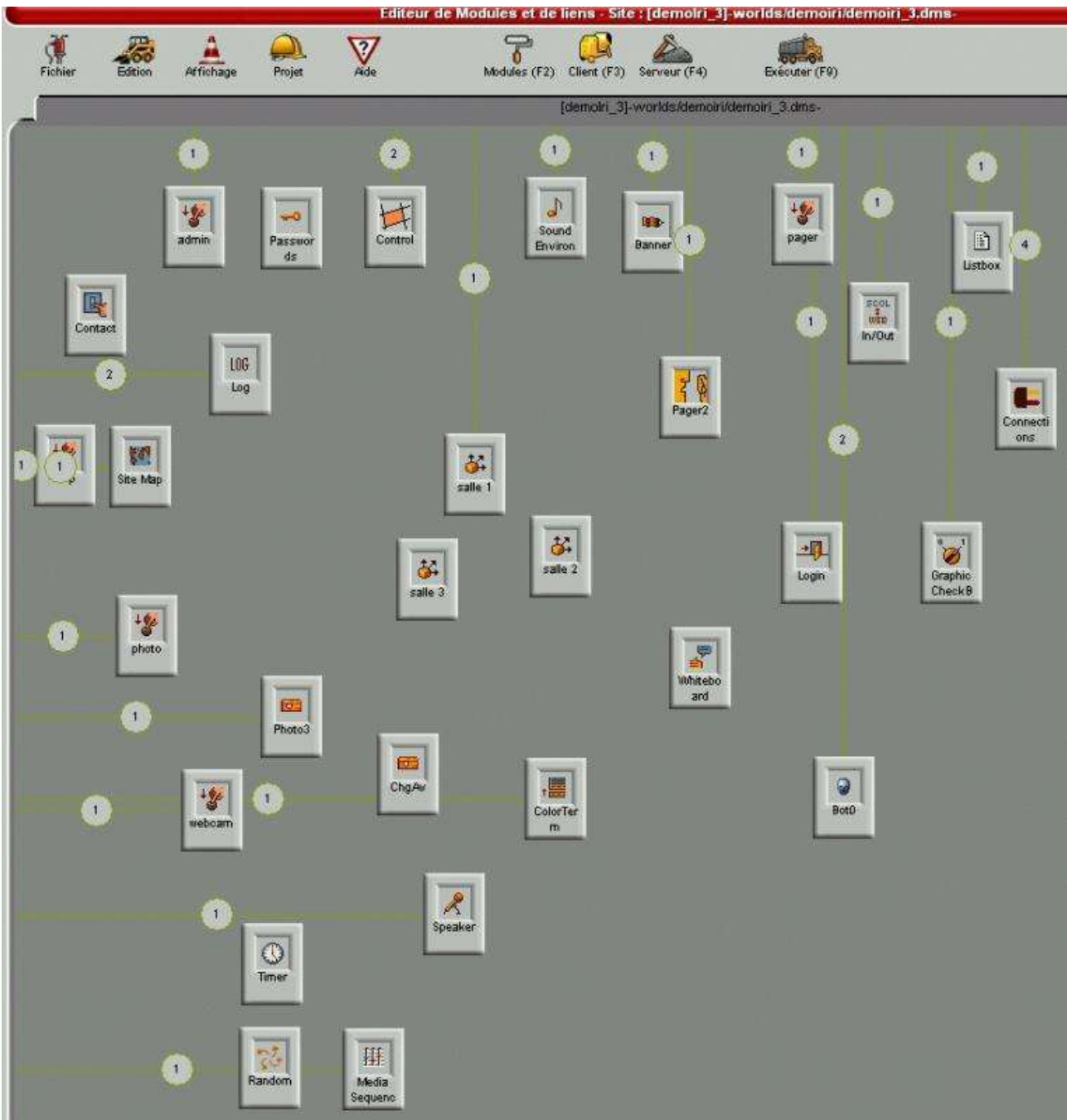
Note : Sa réaction peut également être la diffusion d'une musique, l'ouverture d'une page web, le changement d'une texture de la 3d, etc ... Le module récepteur, l'action et le paramètre seront alors en conséquence.

K/ Final :

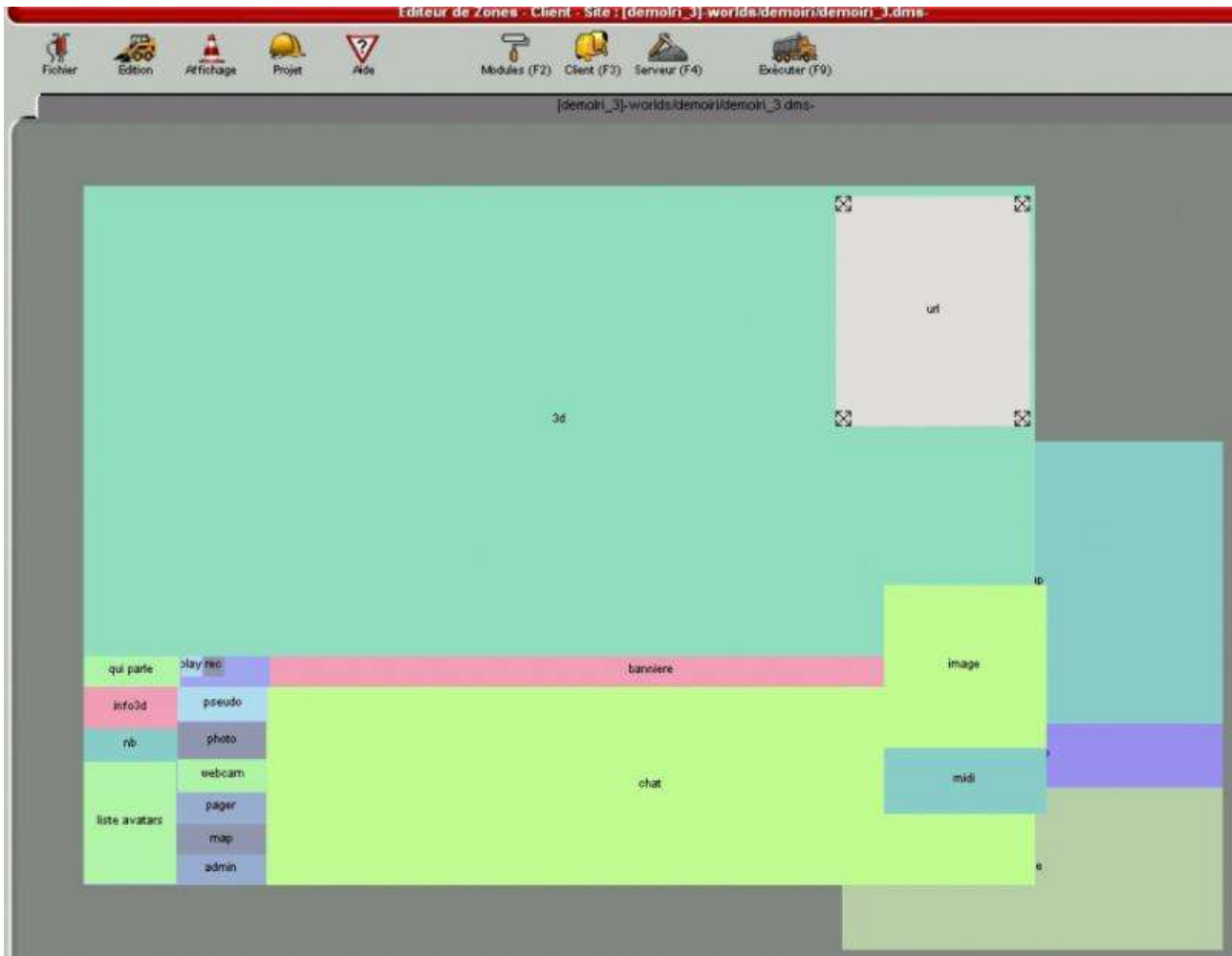
Vous avez terminé ! Je suppose que vous êtes maintenant autonome et que vous vous débrouillez très bien sans filet !

;o)

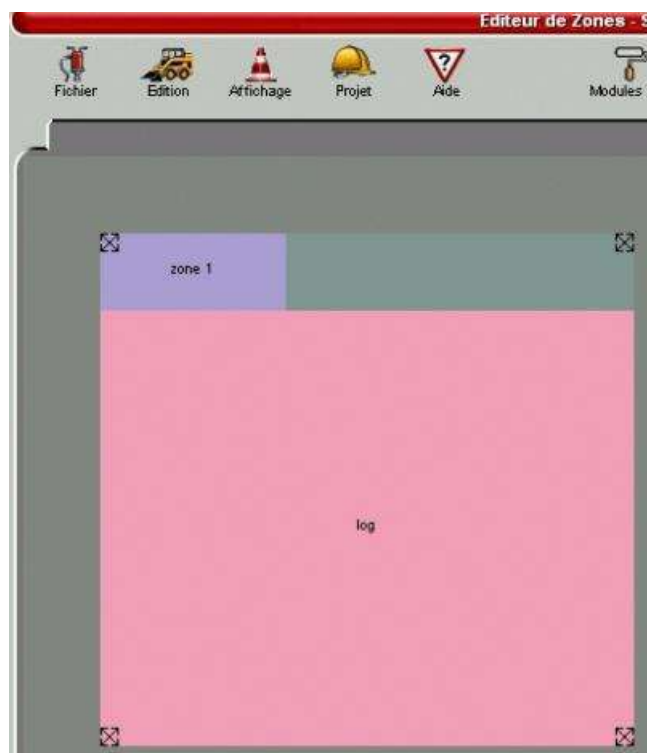
Avant un rapide résumé de ces trois tutoriaux, voici le résultat final de votre apprentissage. La partie Modules (F2) du SCS doit ressembler à ceci :



La partie Cliente (F3) du SCS doit ressembler à ceci :



Enfin, la partie Serveur (F4) du SCS doit correspondre à cela :



Dans les niveaux 1 & 2, nous avons réalisés le site par étapes : l'ajout et la configuration des modules nécessaire à notre projet, l'établissement des liens puis la création des deux interfaces cliente et serveur.

Dans ce tutoriel, nous avons ajouté encore quelques modules et nous avons établi ensuite les liens, et nous avons modifié des modules existants en changeant les liens dans le même temps.

De façon globale, vous avez noté que la plupart des modules se chargent lors de l'ouverture du site (*shell.start* -> *module.start*) ou lors de l'entrée du visiteur dans une cellule 3d (*c3d3.in* -> *module.start*).

Sauf exception, l'événement *module.entering* est généré lorsque le module va être chargé, *module.in* est généré lorsque le module est chargé.

Il n'y a pas de méthodologie précise, mais il est plus facile de garder celle que nous avons suivie. Elle permet de plus d'optimiser la création :

- éviter des modules en doublons (il peut avoir plusieurs moyens pour arriver au même résultat);
- éviter les liens en doublons ou inutiles qui peuvent entrer en conflit ou annuler l'action escompté !;
- éviter des pertes de temps et le fastidieux recommencement lorsqu'on a oublié un élément primordial;
- éviter de charger un module trop tôt ou trop tard (les modules dont on sait que l'utilisation ne se fera qu'après certaines actions du visiteur n'a pas besoin d'être chargé immédiatement);
- éviter de surcharger en plugins et en instances le C3D3 : plus il y en a, plus la scène 3d sera lente (le C3D4, en projet actuellement, devrait résoudre ce point, du moins en partie) en essayant de les utiliser au mieux;
- ...

Autre optimisation à envisager : les textures. Pour les détails, il est inutile d'avoir des fichiers images de 256 x 256 pixels à haute résolution, du 64 x 64 à basse résolution peut être amplement suffisant. Tout comme les niveaux de compression jpeg peuvent être variable. Sauf exception, préférez le jpeg au bmp sauf pour les transparences éventuellement.

Enfin, si vous modélisez vous-même vos scènes 3d, pensez à diminuer le nombre de faces, déjà en supprimant toutes celles qui ne seront pas visibles par le visiteur et en préférant texturer les détails plutôt que les modéliser (cas typique du visage).

Toutes ces optimisations se verront concrètement lors du chargement initial du site et lors de la vitesse d'exécution de chaque rendu.

Bon courage ! :)

Si vous avez des suggestions ou des commentaires à propos de ce tutoriel, faites-moi en part en laissant un message sur mon site.

Si vous avez des difficultés complémentaires, n'hésitez surtout pas à poster un message sur le forum du Scolring (<http://www.scolring.org>).

Iri,