Títol: Granja de Renderitzat per a Blender

Volum: I

Alumne: Pedro Bezerra da Costa

Director/Ponent: Lluís Solano Albajes

Departament: LSI

Data: 21 de Gener de 2008

DADES DEL PROJECTE

Títol del Projecte: Granja de Renderitzat per a Blender

Nom de l'estudiant: Pedro Bezerra da Costa
Titulació: Enginyeria Tècnica en Informàtica de Sistemes
Crèdits: 22,5
Director/Ponent: Lluís Solano Albajes
Departament: LSI

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President: Lluís Pérez Vidal

Vocal: Julián David Morillo Pozo

Secretari: Lluís Solano Albajes

QUALIFICACIÓ

Qualificació numèrica:
Qualificació descriptiva:

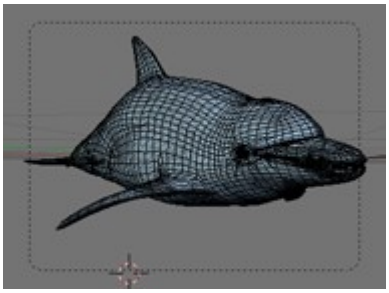
Data:

Índex de contingut

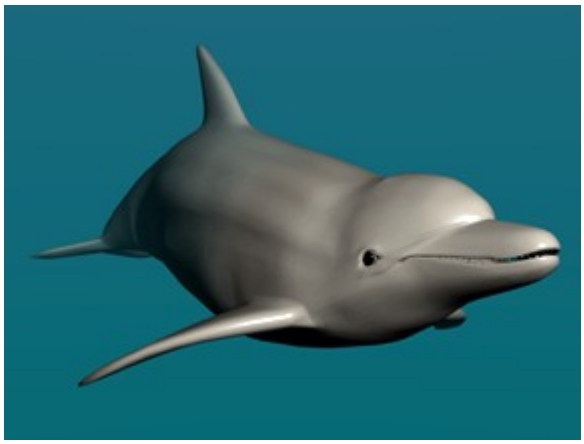
1. Introducció

1.1 *Renderitzar* i el problema del temps

Suposem que un usuari d'una aplicació de gràfics en 3D realitza un gràfic o animació. Per interpretar aquesta informació i mostrar-la com imatge o pel·lícula necessita un següent pas, que és el renderitzat. Aquest procés pot arribar a ser molt costós de processar per la màquina quan volem que la imatge sigui realista, s'han de realitzar molts càlculs de les textures, efectes, il·luminació... etc que conté l'escena del gràfic.



escena 3D



imatge renderitzada

Així, quan ens referim a rederitzar una imatge d'una aplicació 3D com es el cas de Blender parlem del procés de càlcul que fa servir un ordinador per generar una imatge a partir d'una escena 3D. Es poden trigar 7 hores en acabar aquest procés amb una sola màquina de 2Ghz per generar una imatge com la següent.

La solució a aquest problema de temps no és millorar la velocitat del processador, sinó fer servir diversos processadors, cadascun fent una part del treball. En el cas d'una imatge, aquesta la dividim en diverses parts. En el cas d'una animació, la dividim en diversos frames o fotogrames. Repartint el treball a fer entre els diversos ordinadors, aconseguim millorar el temps de renderitzat.

El renderitzat d'imatges es una operació altament paral·lelitzable, ja que cada frame o part de la imatge pot ser calculat independentment de les altres, i després és retornat per ser muntat en la seqüència correcta. Si haguéssim disposat d'una granja amb 4 ordinadors iguals la imatge anterior s'hauria pogut renderitzar en aproximadament 2 hores.

1.2 Com funciona una granja de renderitzat?

Podem definir una granja de renderitzat com un sistema d'ordinadors format per un servidor (màster) i diversos nodes (esclaus) que processen treball en conjunt per tal de representar una o varies imatges en 3D.

El client (usuari):
Envia a la granja l'arxiu amb l'escena en 3D a renderitzar (1). Quan acaben el treball li retornen la imatge o animació (4).

El Servidor (màster):
El servidor rep l'arxiu a renderitzar (1) i reparteix el treball entre els nodes (2). S'encarrega també de controlar la cua de treball, veure quins nodes han acabat i enviar més treball si calgués. Quan els nodes han acabat el treball el servidor retorna les imatges renderitzades al client.

El nodes de treball (esclaus):
Reben l'arxiu en 3D i renderitzen la part que els hi demana el servidor (2). Quan acaben li envien la imatge renderitzada i esperen més treball (3).

Per tal de controlar la granja, hem d'incorporar un *mànager de cues de treball,* que automàticament distribueixi el treball a renderitzar entre els diversos processadors. Aquest treball pot ser el renderitzat d'una única imatge, un grup d'imatges o una secció d'una imatge. Aquest software està organitzat normalment per un servidor i diversos clients que reben i retornen el treball.

1.2.1 Quines aplicacions tenen?

Escoles, despatxos d'arquitectura, estudis d'animació, de disseny o videojocs. Tot usuari que faci servir un programa de disseny 3D i li doni importància al temps i la qualitat del seu treball. El cas amb més èxit de les granges de render han sigut als estudis d'animació de Hollywood, que fan servir granges amb centenars d'ordinadors per poder generar les animacions en un temps que d'altre forma es trigaria anys en aconseguir.

1.3 Descripció del projecte

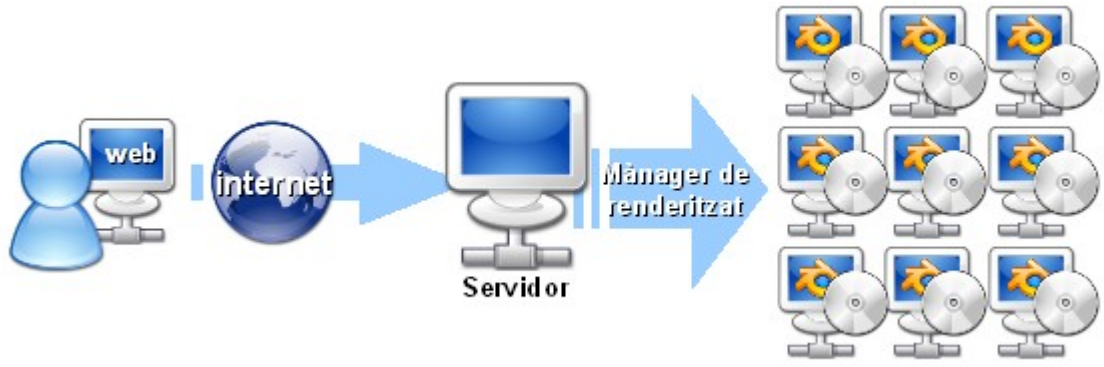
El projecte tracta de veure com muntar i accedir a una granja de renderitzat pel programa de disseny en 3D Blender. Aquest programa es lliure i gratuït, lo que redueix el cost de la granja.

El projecte busca la millor opció per muntar una granja **no permanent** en una empresa o escola que no tingui la opció de comprar més ordinadors que només estiguin dedicats al sistema de renderitzat. Així per exemple, ordinadors que no s'utilitzin a la nit podran convertir-se a nodes de la granja.
L'únic ordinador que cal que tingui una instal·lació permanent es el servidor, els altres son carregats amb un LiveCD.
Mitjançant una aplicació web accedirem a la granja per enviar els treballs al servidor i veurem els arxius renderitzats. També es podrà monitoritzar els treballs que s'estan processant.

Objectius principals

- Muntar un servidor de renderitzat.
- Muntar un sistema operatiu capaç d'executar-se des d'un CD i que actuï com a esclau de la granja de renderitzat.
- Crear una aplicació web per accedir a aquesta granja i poder enviar i rebre treballs per Internet.

L'avantatge d'un sistema com aquest és la possibilitat de poder adaptar-lo a diferents sectors. Es pot crear una granja estàtica i que estigui disponible tot el temps, o una que només funcioni a la nit quan ningú treballa amb les màquines, convertint-les en clients del servidor. L'aplicació web ofereix l'avantatge de disposar de la granja en qualsevol lloc, podent disposar així d'una granja que estigui fora del nostre lloc de treball.



Un usuari accedeix a la granja a través del portal web.

1.4 Motivació

Aquest projecte reuneix tres de les meves principals àrees d'interès al món informàtic. El disseny de pàgines web, el disseny en 3D i l'administració de sistemes.

Després d'haver fet l'assignatura d'animació i disseny en 3D per Blender a l'ETSEIB em vaig interessar per aquesta aplicació de distribució lliure. En fer el treball final em va interessar saber com es podia millorar el temps de render. Llavors vaig començar a investigar sobre diferents solucions que es podrien aportar a una escola o lloc de treball per aprofitar els recursos disponibles, com son els ordinadors que no s'utilitzen o resten apagats durant la nit.

Amb tot això va sorgir la idea de crear una granja de render que es pogués muntar i desmuntar fàcilment amb els ordinadors disponibles d'una escola i que fos accessible des de qualsevol part a través d'Internet.

2. Software

2.1 Debian Linux

Debian GNU/Linux es una distribució GNU/Linux que basa els seus principis en el software lliure. Entre les seves característiques destaquem:

- La disponibilitat en varies plataformes hardware. La versió 4.0 inclou suport per 11 plataformes.
- Una amplia col·lecció de software disponible.
- Un conjunt d'eines que faciliten el procés d'instal·lació i actualització del software.
- No té cap entorn gràfic en especial, el podem deixar en mode de text o instal·lar algun com GNOME, KDE o Xfce.
- Té una gran comunitat d'usuaris i desarrolladors per oferir suport.

2.2 Blender

Blender es un programa multiplataforma, dedicat especialment al modelatge i creació de gràfics tridimensionals. Tot i ser gratuït es molt potent, i és compatible amb Windows, Mac OS X, Linux, Solaris, FreeBSD i IRIX.

2.3 FarmerJoe

Hi havien dues opcions alhora d'escollir un programa lliure i gratuït que controlés la granja.

Nom	*S.O.*	*Llenguatge*	*Renders que suporta*	*A favor*	*En contra*
DrQueue	Linux, MacOS X, Irix, FreeBSD Windows*	C++	Blender Maya Lightwave Mental Ray Softimage XSI	Suporta diversos programes. Té una gran comunitat.	Difícil instal·lació configuració.
FarmerJoe	Linux Windows MacOS	Perl	Blender	Fàcil configuració. Els esclaus no necessiten tenir res instal·lat a part de Blender. Disposa d'una aplicació web per controlar els treballs.	Només suporta Blender.

Llavors perquè escollir Farmerjoe?

Originalment hi havia la idea de fer servir DrQueue, que tot i ser un bon sistema, es força difícil fer-lo funcionar a Windows. Com la meva idea era crear un sistema fàcil d'instal·lar a qualsevol ordinador, DrQueue no m'oferia les característiques ideals ja que s'ha d'instal·lar un programa client a cada maquina que el vol fer servir. Farmerjoe disposa d'un sistema més fàcil i centralitzat, el codi

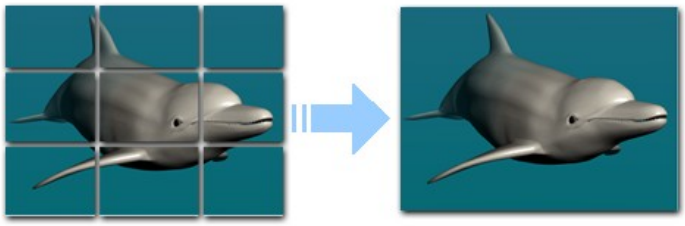
que s'executa en perl queda al servidor màster, per lo que qualsevol canvi només s'ha de fer al servidor. També una raó important era els llenguatges que feia servir. Per crear l'aplicació web m'interessava fer servir Python. Farmerjoe está escrit en perl i disposa d'un script per blender escrit amb python.

En quins sistemes treballa?

En principi esta preparat per Linux, Windows i OSX, però es podria fer servir a qualsevol sistema que pugui executar Perl i Blender.

Característiques

- Permet fer *bucket rendering*: renderitzar una sola imatge, dividint-la en varies parts al clients, per tornar-la a muntar un cop es tenen totes les parts renderitzades.



- Disposa d'un script per enviar treballs des de blender.
- Té un servidor web que et permet veure la cua de treballs.

Els funcionament de Farmerjoe

Farmerjoe funciona amb els següents modes de treball:

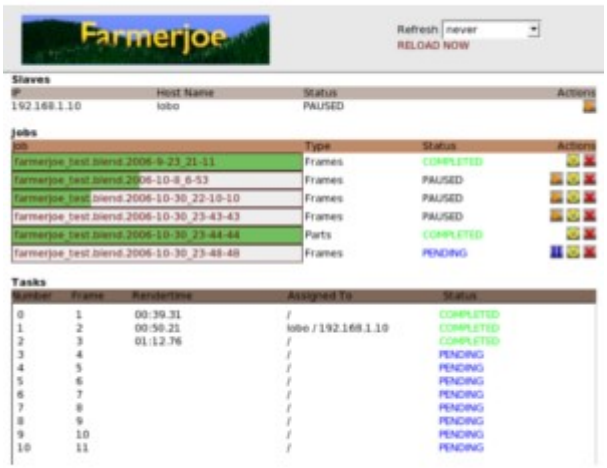
Master (Màster): servidor de treball. Controla la cua de treball i envia la feina als clients de la granja.

Slave (Esclau): client de treball. Rep el treball i ho retorna renderitzat.

Appserver: actua com a servidor d'una pàgina web de monitorització de treballs. Accedint a aquesta pàgina tindrem tot el control dels treballs que s'han enviat:

Jobs: cada treball que s'ha enviat a la granja.

Task: treball assignat a un client-esclau en concret, com un frame o un tros de la imatge a renderitzar.



Appserver de Farmerjoe

Arxius que fa servir

farmerjoe.pl

Programa principal escrit en perl.

farmerjoe.conf

Arxiu amb la configuració de farmerjoe, on s'ha d'incloure la IP del màster, els ports que utilitza l'aplicació i la ruta dels directoris depenent del sistema operatiu.

farmerjoe.state

Inclou tota la informació sobre l'estat dels treballs enviats i els màquines que intervenen.

Directori jobs
El resultat de cada treball queda a la carpeta /frames del directori /jobs.

2.4 LAMP

El terme LAMP és un acrònim que s'origina a finals del 2000 a Alemanya per descriure la plataforma sobre la que funcionen les aplicacions web creades utilitzant la següent combinació d'eines:

- **L**inux com a sistema operatiu;
- **A**pache com a servidor web;
- **M**ySQL com a servidor de bases de dades;
- **P**erl, **P**HP, o **P**ython com a llenguatges de programació.

L'altre opció és WAMP, que és la plataforma equivalent sobre Windows.

2.5 Samba

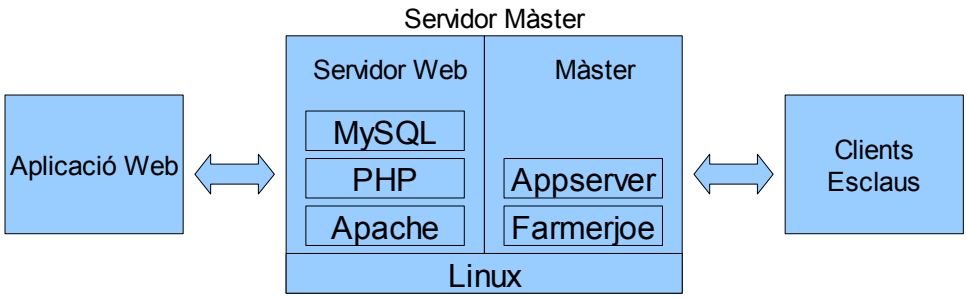
Samba es una implementació lliure del protocol d'arxius compartits de Microsoft Windows (anomenat antigament SMB i recentment CIFS) per sistemes de tipus UNIX. D'aquesta forma, es possible fer-lo servir en ordinadors amb Linux, Mac OSX o Unix en general com a servidors o clients en xarxes de Windows.

Hem escollit samba per compartir fitxers a la nostra xarxa perquè es compatible amb clients amb Windows, així qualsevol ordinador client podrà accedir al servidor de la granja sense preocupar-nos pel seu sistema operatiu.

3. Servidor/Màster

3.1 Funcionament

La màquina que farà de servidor realitza tant les funcions de *màster* de la granja com de servidor web.



3.2 Característiques

El Servidor s'ha muntat amb les següents característiques:

Sistema Operatiu

Linux Debian. Es una de les distribucions més aptes per fer de servidor perquè és molt estable i disposa d'una gran comunitat per oferir suport.

S'ha instal·lat Debian 4.0r1 amb la configuració mínima i després s'han afegit els següents programes:

Gnome: entorn gràfic.
Samba: comparteix la carpeta de farmerjoe.
Perl: permet executar Farmerjoe.
Blender: no es necessari si els clients ho tenen instal·lat.
Farmerjoe: Per distribuir i gestionar els treballs Farmerjoe es carrega en mode *master.* Per a que l'administrador pugui gestionar els treballs s'executa el mode *appserver.*
Imagemagick: Farmerjoe necessita aquest software quan fa servir *bucket rendering* per tornar a unir les parts de les imatges un cop s'han retornat al màster.

Configuració de Apache

S'ha instal.lat la versió d'Apache 2.2.3 (Debian) amb els següents mòduls:

- mod_python/3.2.10
- Python/2.4.4
- PHP/5.2.0-8+etch7
- mod_perl/2.0.2
- Perl/v5.8.8

S'ha configurat la carpeta cgi-bin al directori /usr/lib/cgi-bin

Configuració de Samba

Afegim la carpeta on guardem Farmerjoe al fitxer de configuració de samba:

/etc/samba/smb.conf

```
[global]
      workgroup = Mshome
      passdb backend = tdbsam
      obey pam restrictions = yes
      server string = slax
      restrict anonymous = no
      domain master = no
      preferred master = no
      max protocol = NT
      acl compatibility = winnt
      ldap ssl = No
      server signing = Auto
      security = share
      passwd chat debug = yes
      passwd chat timeout = 10
      disable netbios = yes

[render]
      path = /render
      guest ok = yes
      public = yes
      avaiable = yes
      browseable = yes
      writable = yes
```

Configuració de Perl

Son necessàries les següents llibreries per Farmerjoe:

Tiny.pm

Zlib.pm

YAML

Configuració dels programes d'inici

Per afegir Farmerjoe als programes que s'arrancaran a l'inici.

A la carpeta /etc/init.d/ es crea el següent script:

```
# vim /etc/init.d/farmerjoe.sh
```

Amb les comandes següents:

```
perl /render/Farmerjoe2.pl --master &
perl /render/Farmerjoe2.pl --appserver &
```

I s'actualitza l'script d'inici.

```
# update-rc.d farmerjoe.sh defaults 19
```

4. Clients/Esclaus

4.1. Funcionament

Qualsevol màquina que tingui accés al Servidor Master, porti instal·lat Blender i perl podrà fer d'esclau. Un dels objectius del treball ha sigut crear una distribució per a que qualsevol ordinador de la xarxa local es converteixi en esclau sense haver de configurar-lo ni modificar el sistema operatiu que ja tingui instal·lat. Com la majoria de màquines son PC's amb processador 386 crearem una distribució per aquest tipus de màquina.

4.2 Distribució LiveSlave

4.2.1 Que és un LiveCD?

Un **LiveCD, LiveDVD** o **LiveDistro**, es un sistema operatiu, normalment acompanyat d'un conjunt de programes i que es guarda en un medi extraïble, sigui un CD (d'aquí ve el seu nom) o un DVD que s'executa des d'aquest sense necessitar d'instal·lació al disc dur de l'ordinador, per lo que utilitza la memòria RAM com a disc dur virtual i el propi medi com a sistema de fitxers. Això permet deixar la computadora intacta un cop acabada la feina. Per fer-lo servir s'ha de configurar l'arranc de l'ordinador per a que agafi la unitat lectora CD.

Es diu **LiveUSB** quan enlloc d'un CD fem servir una memòria USB.

4.2.2 Característiques del nostre LiveSlave

El LiveCD s'ha creat amb les següents característiques:

Sistema Operatiu: Linux Debian 4.0 estable. Aquest sistema operatiu s'ha configurat de forma que carregui el programa FarmerJoe en mode *slave* i reconegui el servidor automàticament.

Software

Samba: es connectarà a la carpeta compartida del servidor.
Perl: en permetrà executar el script de Farmerjoe que resideix al servidor.
Blender: per no carregar de peticions el servidor executarem blender des de la propia distribució.

4.2.3 Configuració de la distribució

Per instal·lar Debian 4.0r1 amb la configuració mínima i sense entorn gràfic baixem de la pagina de Debian la següent imatge iso de CD:

debian-40r1-i386-netinst.iso

Que et permet instal·lar els paquets bàsics per fer funcionar el sistema operatiu. Després anirem actualitzant el sistema a través d'una connexió d'Internet i instal·larem els paquets de repositoris apt.

Per instal·lar samba:

```
# apt-get install samba-common
# apt-get install samba-client
# apt-get install smbfs
```

Per instal·lar Perl:

```
# apt-get install perl
# apt-get install libyaml-perl
# apt-get install libio-zlib-perl
```

Instal·lar llibreries perl Tiny.pm i Zlib.pm necessàries per Farmerjoe.

```
mkdir /user/lib/perl/5.8/YAML
mkdir /user/lib/perl/5.8/Compress
mv Tiny.pm /usr/lib/perl/5.8/YAML/
mv Zlib.pm /usr/lib/perl/5.8/Compress/
```

Per instal·lar Blender:

```
# apt-get install blender
```

Crearem el directori on es muntará Farmerjoe.

```
# mkdir /render
```

Per configurar els programes que s'arrancaran a l'inici.
A la carpeta /etc/init.d/ es crea el següent script:

```
# vim /etc/init.d/farmerjoe.sh
```

Amb les comandes següents:

```
dhclient
sleep 10
mount -t smbfs -o username=<usuari>, passwd=<password>, dmask=777,
fmask=777 //<ip de master>/render /render
perl /render/Farmerjoe2.pl –slave &
```

Amb això tenim una distribució linux que en iniciarse es connecta automàticament al servidor i executa Farmerjoe en mode slave.

4.2.4 Creant un LiveCD amb Linux Live Scripts

Linux Live es una serie de shell scripts que et permeten crear una distribució de Linux autoarrancable a partir d'una distribució Linux que tinguem instal·lada. Aquest Live System que crearem serà bootable des de CD-ROM o disc portable, com una memòria flash o un llapis de memòria USB.

Descarreguem de la pàgina http://www.linux-live.org/ els Linux Live Scripts i el kernel precompilat amb els mòduls aufs i squashfs.

1. Canviem el kernel per un que disposi dels moduls aufs, squashfs+LZMA.
2. S' eliminen tots els arxius innecessaris, per exemple els manuals, per fer el nostre Live CD lo més petit possible.
3. Descomprimim Linux Live Scripts a /tmp

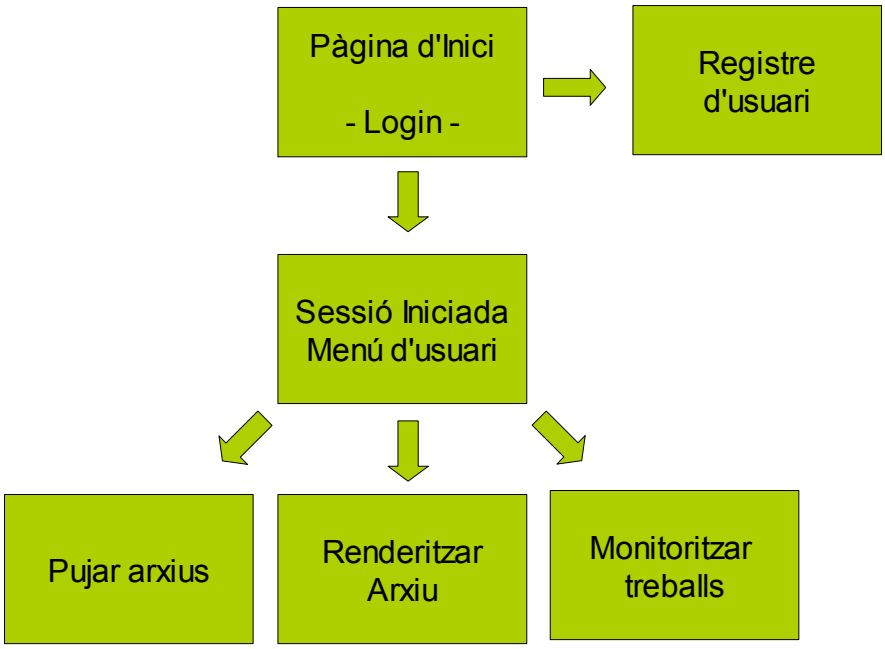
   Executem ./build

La distribució serà creada a /tmp/live_data_1234 on 1234 es un número a l'atzar.

4. Finalment, per crear la imatge ISO executem make_iso.sh
   Si volguéssim crear un disc USB bootable executem bootinst.sh

5. Aplicació Web

5.1 Esquema de funcionament



5.2 Llenguatges utilitzats

HTML i PHP

HTML és el llenguatge de marcació per a estructurar textos i presentar-los en el format estàndard de les pàgines web. PHP és un llenguatge de programació que es fa servir normalment per a la creació de contingut per a documents HTML. El document es interpretat al servidor i aquest pot retornar una pàgina en HTML al client. Utilitzarem PHP a la nostra aplicació web per generar contingut dinàmic.

CSS

CSS (Cascading Style Sheets) és un llenguatge formal per a definir la presentació d'un document estructurat en HTML o XML. Permet definir diversos aspectes de tots els elements que componen les pàgines web i l'utilitzarem per definir l'aspecte visual de l'aplicació web.

SQL

SQL (Structured Query Language) és un llenguatge declaratiu per a accedir a bases de dades relacionals.

La seva utilització ha estat per realitzar la identificació d'usuaris i emmagatzemar la seva configuració a una base de dades MySql, que també és de lliure distribució.

Ha estat un ús molt "lleuger" ja que aquestes bases de dades permeten realitzar tasques molt més complexes, però ens ha permès aprendre els conceptes bàsics i com relacionar-les amb aplicacions web per mitjà de PHP.

XML

XML no és un llenguatge en particular, sinó una forma de definir llenguatges que permet definir els seus propis elements. La seva feina principal es facilitar la compartició d'informació entre sistemes i aplicacions.

YAML

Es una "versió de xml" que resulta més fàcil d'escriure. El canvi de fer servir YAML enlloc de XML per guardar l'estat dels treballs a farmerjoe suposa un fitxer 3.5 vegades menor.

Python

Llenguatge de programació d'alt nivell, dissenyat per donar preferència a la intel·ligibilitat i la importància del programador enlloc de la feina de l'ordinador a partir d'una sintaxi i semàntica minimalista. És molt utilitzat en aplicacions web.

Perl

Perl es un llenguatge de propòsit general originalment creat per manipular text i utilitzat ara per un ampli ventall de tasques com l'administració de sistemes o la creació de webs. El llenguatge intenta ser pràctic, eficient i complet més que bonic, elegant i minimalista.

5.3 Joomla

Qué és Joomla?

Joomla és un sistema gestor de continguts dinàmics (CMS o Content Management System) que permet crear llocs web d'alta interactivitat, professionalitat i eficiència. És una solució de codi obert que està disponible de manera gratuïta per a tothom. Pot ser emprat per a administrar fàcilment qualsevol aspecte d'un lloc web. Per instal·lar-lo es necessita un servidor web que suporti PHP i disposi d'una base de dades MySQL.

Farem servir aquest gestor de continguts per registrar i donar accés als usuaris d'una forma segura per a accedir a la granja de renderitzat.

Es necessari crear un component per a aquest gestor per dotar-lo de les funcionalitats necessàries per pujar, renderitzar i mostrar els treballs de la granja de renderitzat.

Tipus d'usuari

Joomla disposa dels diversos tipus d'usuari que definim segons els permisos que tinguin i s'agrupen entre els que només poden accedir a la visualització de navegació (front-end), i els que poden accedir al sistema d'administració (back-end).

El sistema d'administració s'accedeix a partir de la pàgina http://<servidor>/administrator

Plantilles

Les plantilles defineixen la disposició del contingut a la pàgina. Juntament amb CSS donen forma al contingut de la web. Per la nostra web s'ha fet servir com a base una plantilla lliure de www.siteground.com i s'ha modificat.

La forma que tenen de mostrar el contingut es a través de crides en PHP com la següent:

```
<?php mosMainBody(); ?>
```

Que mostra el cos principal de la pàgina.

Components

Els Components son elements que donen noves funcionalitats a Joomla i es mostren en el cos principal de la plantilla de la pàgina web.

5.4 El component FarmJobs

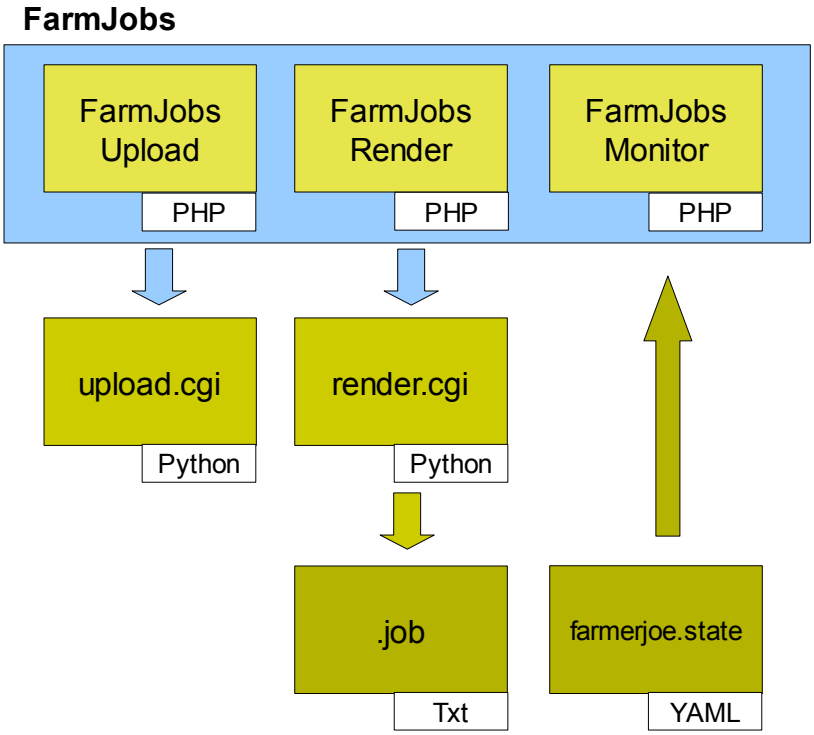
Per donar a Joomla les funcions necessàries per accedir a la granja, hem creat el següent component que anomenarem FarmJobs (treballs de granger). Aquest tindrà tres tasques principals:

FarmJobs Upload: pujarem els fitxers a partir d'aquesta pàgina.

FarmJobs Render: ens permet escollir quin dels fitxers que hem enviat volem renderitzar.

FarmJobs Monitor: podem veure els treballs que s'han enviat a renderitzar.

5.4.1 Esquema de funcionament



5.4.2 Arxius de FarmJobs

Arxius propis del component:

farmjobs.xml – Arxiu d'instal·lació del component.

farmjobs.php – Mostra la informació al frontend.

admin.farmjobs.php – Realitza les crides a la base de dades i configura la sortida HTML.

admin.farmjobs.html.php – Controla les sortides.

install.farmjobs.php – Arxiu d'instal·lació.

uninstall.farmjobs.php – Arxiu de desinstal·lació.

toolbar.farmjobs.php – Configura la barra del menú a la zona d'administració.

toolbar.farmjobs.html.php – Controla el que mostra la barra de menú.

sfYaml.class.php – La classe sfYaml permet processar de forma senzilla qualsevol arxiu en format YAML. Es tracta d'un processador (parser) d'arxius YAML que els converteix en vectors associatius de PHP.

Spyc.class.php - Spyc carrega i descarrega documents en YAML. Necessari per la classe sfYaml.

TextHelper.php – Requerit per sfYaml.class.php

Arxius que accedeix el component a la carpeta cgi-bin:

upload.cgi – Escrit en python, guarda el document del formulari de render al servidor i carrega la funció per crear el treball a l'arxiu job.py amb les dades del formulari.

render.cgi – Escrit en python, genera un nou treball amb l'arxiu blender i els paràmetres enviats per l'usuari.

5.4.3 FarmJobs Upload

Conté un formulari on es demana quin arxiu vol pujar. Utilitza upload.cgi per fer la operació.

upload.cgi

Puja el fitxer seleccionat al directori de l'usuari. No permet fer la pujada en els següents casos:

- Si l'arxiu no té l'extensió de Blender es cancel·la l'operació i dona un avís.
- Si la quota de disc es sobrepassa.

5.4.4 FarmJobs Render

L'usuari pot escollir des d'una llista desplegable amb els arxius que ha pujat al servidor, quin vol renderitzar. A més, se li demanen els paràmetres següents:

Frame inici: El primer fotograma de l'animació.

Frame final: L'últim fotograma.

Render Step: El nombre de imatges que cada esclau renderitza en una única passada.

Timeout (mins): els minuts que han de passar fins donar la imatge a renderitzar per perduda.

Si el Frame inici coincideix amb el Frame final, la imatge es renderitza amb *bucket rendering*. Es divideix en diverses parts que s'envien a renderitzar separadament als esclaus.

En un primer moment es pensava incloure la resolució de la imatge, però blender no accepta aquest paràmetre en mode de línia de comandes.

render.cgi

Aquest programa en llenguatge Python prepara els directoris, copia l'arxiu de blender seleccionat de la carpeta uploads i crea l'arxiu .job amb els paràmetres del formulari. Realitza una crida a Farmerjoe indicant el treball enviat.

Exemple de treball *arxiu*.job

Quan executem render.cgi aquest genera un arxiu de treball per la granja amb el següents paràmetres:

```
crystal_cube.blend                          #nom de fitxer
2                                           #frame d'inici
2                                           #frame final
2                                           #step
12000                                       #timeout
```

```
crystal_cube.blend.2008-1-8_23-54-54      #directori de treball
crystal_cube.blend                        #nom del treball
800                                       #amplada, no es fa servir
600                                       #alçada, no es fa servir
1                                         #divisions en horitzontal
1                                         #divisions en vertical
admin (nom de l'usuari del treball)
```

5.4.5 FarmJobs Monitor

Slaves

IP	Host Name	Status
192.168.1.2	master.local	IDLE
192.168.1.4		BUSY
192.168.1.5		IDLE

Jobs

Job	Type	Frames	Status
monkey.blend.2008-1-13_13-20-20	Frames	1/1	COMPLETED
teapot_soft.blend.2008-1-13_18-9-9	Parts	71/1	ACTIVE

Una primera taula ens mostra les màquines esclau que actualment tenim connectades al servidor. Ens indica la seva direcció IP, el nom de la màquina i el seu estat, que pot ser IDLE (en espera de rebre treballs), BUSY (està processant treball) o PAUSE (aturada per l'administrador).

La segona taula mostra la llista de treballs que el nostre usuari ha enviat a renderitzar. El nom del treball (Job), conté un link on podrem veure i descarregar les imatges que s'han renderitzat. Després ens mostra el tipus de renderitzat que pot ser en diferents frames o parts (d'una sola imatge), els frames que s'han completat i l'estat que pot ser ACTIVE (queden frames per renderitzar), COMPLETED o en PAUSE.

Per llegir aquestes dades hem utilitzat el fitxer en format YAML farmerjoe.state creat per l'aplicació farmerjoe i que accedim de la següent manera.

Exemple de farmerjoe.state:

```
---
clients:
  5:
    hostname: master.local
    id: 5
    ip: 192.168.1.2
  6:
    hostname: master.local
    id: 6
    ip: 192.168.1.3
    status: IDLE
    type: SLAVE
id: farmerjoe
jobs:
  -
    blendfile: crystal_cube.blend
    endframe: 2
    id: 1199832874
    image_x: 800
    image_y: 600
    jobdir: crystal_cube.blend.2008-1-8_23-54-54
    jobfile: crystal_cube.blend.
2008-1-8_23-54-54/crystal_cube.blend.job
    jobname: crystal_cube.blend
    startframe: 2
    status: ACTIVE
    step: 2
    tasks:
      -
        allocated: 0
        assigned_to: 6
        frame: 2
        number: 0
        start_time: 1200226914
        status: RENDERING
    timeout: 12000
    type: Frames
    user: pidrosoft
    xparts: 1
    yparts: 1
```

Codi php amb una consulta que mostra els cliets de tipus *slave* :

```
//carreguem la classe yaml
require "sfYaml.class.php";


//recollim el fitxer yaml
$my_yaml = sfYaml::load('/render/farmerjoe.state');

//mostrem els esclaus dins d'una taula que ja ha estat creada
foreach ($my_yaml[clients] as $slaves) {
    if($slaves[type] == "SLAVE"){
      echo "<tr>";
      echo "<td>".$slaves[ip]."</td>\n";
      echo "<td>".$slaves[hostname]."</td>\n";
      echo "<td>".$slaves[status]."</td>\n";
      echo "</tr>\n";
    }
}
```

5.4.6 Modificant Farmerjoe.pl

Farmerjoe s'ha modificat per crear els renderitzats en el directori de cada usuari, ja que abans guardava tots els resultats en una mateixa carpeta.

Hem modificat les següents linies en perl, afegint el paràmetre -o per indicar a blender en quin directori ha de guardar els arxius renderitzats.

Línia 227:

```
my $args = " -b".'"'.$self->{c}{jobs}.$self->{c}{sep}.$c-
>{blend}.'"'." -o //frames/ -s ".$c->{start}." -e ".$c->{end}.$anim;
```

Línia 307:

```
my $args = " -b ".'"'.$self->{c}{jobs}.$self->{c}{sep}.$c-
>{blend}.'"'." -o //frames/ -s ".$c->{frame}." -e ".$c->{frame}." -P
".$self->{c}{bucket_render};
```

Línia 347:

```
my $args = " -b ".'"'.$self->{c}{jobs}.$self->{c}{sep}.$c-
>{blend}.'"'." -o //frames/ -s ".$c->{start}." -e ".$c->{end}.$anim;
```

També s'ha modificat la forma en que crea l'arxiu farmerjoe.state, afegint com a nou paràmetre el nom d'usuari, que haviem creat al arxiu .job del treball quan enviem l'arxiu ha renderitzar.

Les següents línies han estat incorporades al codi:
Línia 482:

```
$job_data->{user} = $job_data_file->[11];
```

5.4.7 Instal·lació del Component

A l'àrea d'administració en el menú d'instal·ladors tenim la opció d'instal·lar nous components.

Després d'instal·lar el component, s'han creat els enllaços a les diferents tasques del component en el menú d'usuari.

FarmJobs Upload

```
index.php?option=com_farmjobs&task=upload
```

FarmJobs Render

```
index.php?option=com_farmjobs&task=render
```

FarmJobs Monitor

```
index.php?option=com_farmjobs&task=monitor
```

6. Temps i Cost

6.1. Temps dedicat

Treball d'investigació i proves sobre diferents distribucions de Linux.....**40 hores**

Proves amb software de renderitzat (Dr.Queue, Farmerjoe) .....**48 hores**

Configuració de la granja:

Configuració del Servidor Màster.....48 hores

Configuració del Client Esclau .....40 hores

Creació de la distribució LiveSlave .....24 hores

Total .....**112 hores**

Disseny i implementació de l'aplicació web:

Instal·lació de Joomla + Plantilla .....8 hores

Implementació de upload.cgi .....64 hores

Implementació de render.cgi .....64 hores

Creació de component per Joomla .....128 hores

Total .....**264 hores**

Total del projecte .....**464 hores**

6.2. Cost del projecte

Software:

Blender .....0 euros

Farmerjoe .....0 euros

Sistema LAMP .....0 euros

Hardware:

El projecte aprofita els recursos de l'empresa o escola per crear la granja, lo que no suposa un cost addicional. Tot i així el cost del servidor que s'hauria de comprar en cas de no disposar de cap maquina lliure variaria depenent del nombre de màquines que li volguéssim connectar i del tipus de feina.

Servidor .....1000 – 3000 euros

Suposant que estiguéssim cobrant 30 euros/hora pel treball d'enginyer tècnic de sistemes.

Personal .....13.920 euros

7. Valoració Final

7.1 Conclusions

Els objectius principals s'han acomplert satisfactòriament. El gran repte d'aquest projecte ha sigut entendre el funcionament de Farmerjoe, la principal aplicació al voltant de la qual tot havia d'anar muntat. Farmerjoe va per la versió 0.1.3 en aquest moment, i no disposa d'una comunitat al seu voltant com es el cas de Dr.Queue, però crec que ha sigut la millor opció per obtenir els resultats que volia.

7.2. Millores del projecte

Per falta de temps no s'han arribat a implementat les següents funcions al component FarmJobs.

- Poder escollir el format de la imatge a renderitzar.
- Poder enviar arxius comprimits.
- Poder aturar o cancel·lar treballs des de FarmJobs Monitor.
- Poder esborrar els arxius enviats.

Les actualitzacions d'aquest projecte estaran disponibles a http://code.google.com/p/farmjobs/

7.3 Valoració personal

Amb aquest projecte he aconseguit millorar els meu nivell com administrador de sistemes, concretament en sistemes Linux, entenent millor com funciona, com es configura i les eines que fa servir. He aprés nous conceptes i aclarit els que ja havia adquirit en la carrera.

També he ampliat els meus coneixements de programació aprenent dos nous llenguatges, Python i PHP, i apropant-me a Perl. Entre moltes coses, m'han permès veure com intercanviar informació entre els diferents llenguatges o fitxers. Totes aquestes tecnologies em seran molt útils, ja que son molt utilitzades per crear continguts web dinàmics.

En general a sigut un treball engrescador que m'ha fet donar un pas enrere molts cops, per poder donar un salt endavant uns altres...

Referències

Pàgines Web

Blender.org
http://www.blender.org/

Debian.org
http://www.debian.org/

Farmerjoe
http://blender.formworks.co.nz/

Dr.Queue.org
http://drqueue.org/cwebsite/

Linux Live Scripts
http://www.linux-live.org/

Syntax Error: Montar Samba
http://www.syntaxerror.es/2007/06/15/montar-samba-en-ubuntu-festy-server-carpetas-compartidas/

Apache2 Configuration
http://www.mneylon.com/blog/archives/2004/07/25/apache-2-configuration/

Programming CGI With Python
http://python.about.com/od/cgiformswithpython/ss/pycgitut1_2.htm

Voidspace CGI Applications
http://www.voidspace.org.uk/python/cgi.shtml

Python Interactive CGI Tutorial
http://www.cs.virginia.edu/~lab2q/lesson_1/

Creating a quick and easy Mambo component, Doyle Lewis
http://www.mambohut.com/content/view/158/82/

Joomla Forum
http://forum.joomla.org/index.php?topic=109473.msg1014225

Spyc, a simple php yaml class
http://spyc.sourceforge.net/

Zlib, lliberia per Python
http://search.cpan.org/~pmqs/Compress-Zlib-2.008/lib/Compress/Zlib.pm

Bibliografia

Python in a Nutshell, Alex Martelli. Ed. O'Reilly 2003.

Annex

9.1 Instal·lació de Farmerjoe al Servidor

Farem servir els arxius disponibles al DVD adjunt al projecte.

9.1.1 Requisits previs

- Perl
- Llibreries:
  - Tiny.pm
  - Zlib.pm
  - YAML

1. Descomprimir l'arxiu “farmerjoe_we.zip” a la carpeta /render al directori arrel del sistema.
2. Crear els directoris /render/jobs i /render/uploads
3. Donar els permisos als directoris
4. Compartir la carpeta /render amb Samba.

9.1.2 Editar l'arxiu de configuració Farmerjoe.conf

- Editar l'adreça IP del servidor màster i el número del port (per defecte 2006) que farà servir l'aplicació.
- Editar el directori on es troba compartit Farmerjoe. En linux fem servir /render, i a Windows es pot mapejar a la unitat de xarxa r:
- Editar la ruta per accedir a blender.

```
# Master Server Configuration
port = 2006
master = 192.168.1.2

#carpeta de treballs
jobs = jobs

#carpeta de logs
logs = logs

linux_root = /render
linux_blender = /render/blender
linux_composite = /usr/bin/composite

windows_root = r:
windows_blender = r:\bin\windows\blender\Blender.exe
windows_composite = composite
```

```
osx_root = /Volumes/farmerjoe
osx_blender
= /Volumes/farmerjoe/bin/osx/blender/blender.app/Contents/MacOS/blende
r
osx_composite = /usr/local/bin/composite


# Application server Configuration
appserver_port = 2007
```

9.2 Instal·lació de l'aplicació Web

Aquí no indicarem com fer una instal·lació d'inici de Joomla, que podem trobar a http://ayuda.joomlaspanish.org/ sinó que instal·larem la pàgina web des del DVD amb els components i continguts ja configurats.

9.2.1 Requisits previs

Aquests son els requisits mínims necessaris abans d'instal.lar Joomla! 1.0.13.

- PHP 4.2.x o superior - http://php.net
- MySQL 3.23.x o superior - http://mysql.org
- Apache 1.3.x o superior - http://httpd.apache.org

També has d'assegurar tenir les funcionalitats de MySQL, XML i Zlib activades a l'instal.lació de PHP. A partir d'aquí assumirem que disposem d'un servidor funcionant completament.

9.2.2 Preparar la base de dades

Es necessari tenir una base de dades MySQL abans d'iniciar el procés d'instal·lació. En el nostre cas crearem la base de dades amb el nom “farm”, l'usuari “farmer” i el password “2182bf”. Restaurem la base de dades a partir del backup “farm_2008-01-20_19-08.sql”

9.2.3 Instal·lació Joomla

Descomprimim l'arxiu “Joomla_1.0.13-farmjobs.zip**”** en un directori arrel del servidor web. En el nostre cas aquest serà /var/www/bf.

1. Crear la carpeta /jobs amb un enllaç feble a /render/jobs
2. Crear la carpeta /uploads amb un enllaç feble a /render/uploads

Després editem l'arxiu configuration.php amb la informació del nostre servidor:

configuration.php

Directori on tenim la pàgina:

```
$mosConfig_absolute_path = '/var/www/bf';
```

Directori cache:

```
$mosConfig_cachepath = '/var/www/bf/cache';
```

Nom de la base de dades:

```
$mosConfig_db = 'farm';
```

Direcció de la pàgina “online”

```
$mosConfig_live_site = 'http://domini/bf';
```

Mail de l'administrador:

```
$mosConfig_mailfrom = 'mail@domini.com';
```

Nom d'usuari de la base de dades:

```
$mosConfig_user = 'usuari bbdd';
```

9.2.4 El component farmjobs

Amb la instal·lació el component ja ve instal·lat, però si li volem fer modificacions disposem de l'arxiu “com_farmjobs.zip” que haurem de reinstal·lar des del menú “instaladores” del panel de control d'administració.

9.2.5 Cgi-bin

Al directori cgi-bin que tinguem configurat per apache copiarem els següents arxius i aplicarem permisos d'execució:

- render.cgi
- upload.cgi

9.3 Manual d'usuari

9.3.1 Per administrar la pàgina web:

1. Accedim al panell d'administrador a “http://domini/bf/administrator”
2. Entrem com a login “admin” i el password “2182bf”. Després canviarem el password al menú “Sitio//Administrador de Usuarios”. Des d'aquest menú també controlem tots els usuaris registrats i quins permisos d'accés tenen.

9.3.2 Per monitoritzar els treballs de la granja

1. Accedim com administrador al login de la pàgina principal “http://domini/bf/”.
2. Entrem a l'enllaç Monitoritzar treballs.
    1. Podem veure la llista d'esclaus i aturar-los amb el botó de pausa.
    2. Podem veure la llista de treballs i resetejar-los o cancel·lar-los amb els botons que es troben al seu costat.

9.3.3 Per crear un nou “Esclau” amb LiveSlave CD:

1. Introduïm un CD creat amb la imatge “liveslave.iso” i anem a la configuració de boot del PC.
2. Fem que arranqui des de CD com a primera opció. Guardem els canvis.

3. Quan el sistema arrenqui no cal entrar el login, els programes ja estaran funcionant.

9.3.3 Per crear un nou “Esclau” amb Windows:

1. Tenir instal·lat Perl amb les llibreries YAML::Tiny i Compress::Zlib.
2. Tenir instal·lat Blender.
3. Accedir a la carpeta compartida /render del servidor i muntar-la a la unitat r:
4. Executar amb Perl l'arxiu r:\farmerjoe.pl