Numero 16 - Marzo 2013

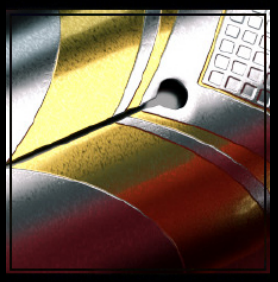
BLENDER
Magazine Italia

SCARECROW
Making of dell'immagine vincitrice
del contest di Blendercoockies

FOTOINSERIMENTI SEMPLICI
Impariamo ad usare BLAM
per gli inserimenti fotografici

TUTORIAL
"Script in Python"
e "Uva realistica"

Indice



Editoriale

Blender Magazine Italia
numero 16
anno 2013

Responsabili:
Luca Pinciani (Sinistar)
Alfonso Annarumma (Anfeo)

Collaboratori:
Andrea Fiocca (gikkio)

Grafica:
Davide_G

Siti di riferimento:
www.blender.it
www.kino3d.com
www.blender.org

Software utilizzati:
Blender
Scribus
Pdftk
The Gimp
LibreOffice

Visibilità

Spopola in questi giorni la protesta e la solidarietà verso i VFX Artist sfruttati e malpagati dai colossi del cinema. Sarò forse l'unico ad essere poco sorpreso di questa faccenda?
Qua in Italia subiamo lo stesso maltrattamento da sempre e chiunque si sia occupato in vita sua di qualche lavoro di Grafica lo sa ampiamente.
La polemica che condivido ampiamente non è legata agli esigui compensi (o del tutto assenti in certi casi) dovuti a gare al ribasso fatte dagli Studi di VFX per accaparrarsi il lavoro. Le conseguenze di queste azioni sono dovute a scelte sbagliate di chi gestisce questi studi.
Sono d'accordo, invece, sulla poca importanza che viene data ai VFX. Ad esempio, il nome di un direttore degli VFX di un un film dovrebbe comparire subito dopo quello della produzione, almeno in certi film dove il 70% delle scene sono realizzate (o ritoccate) in CG.
Un film di fantascienza qualsiasi degli ultimi 10 anni (ma anche di più) dovrebbe essere presentato come: film di Regista, prodotto da Produzione, VFX di Studio di VFX.

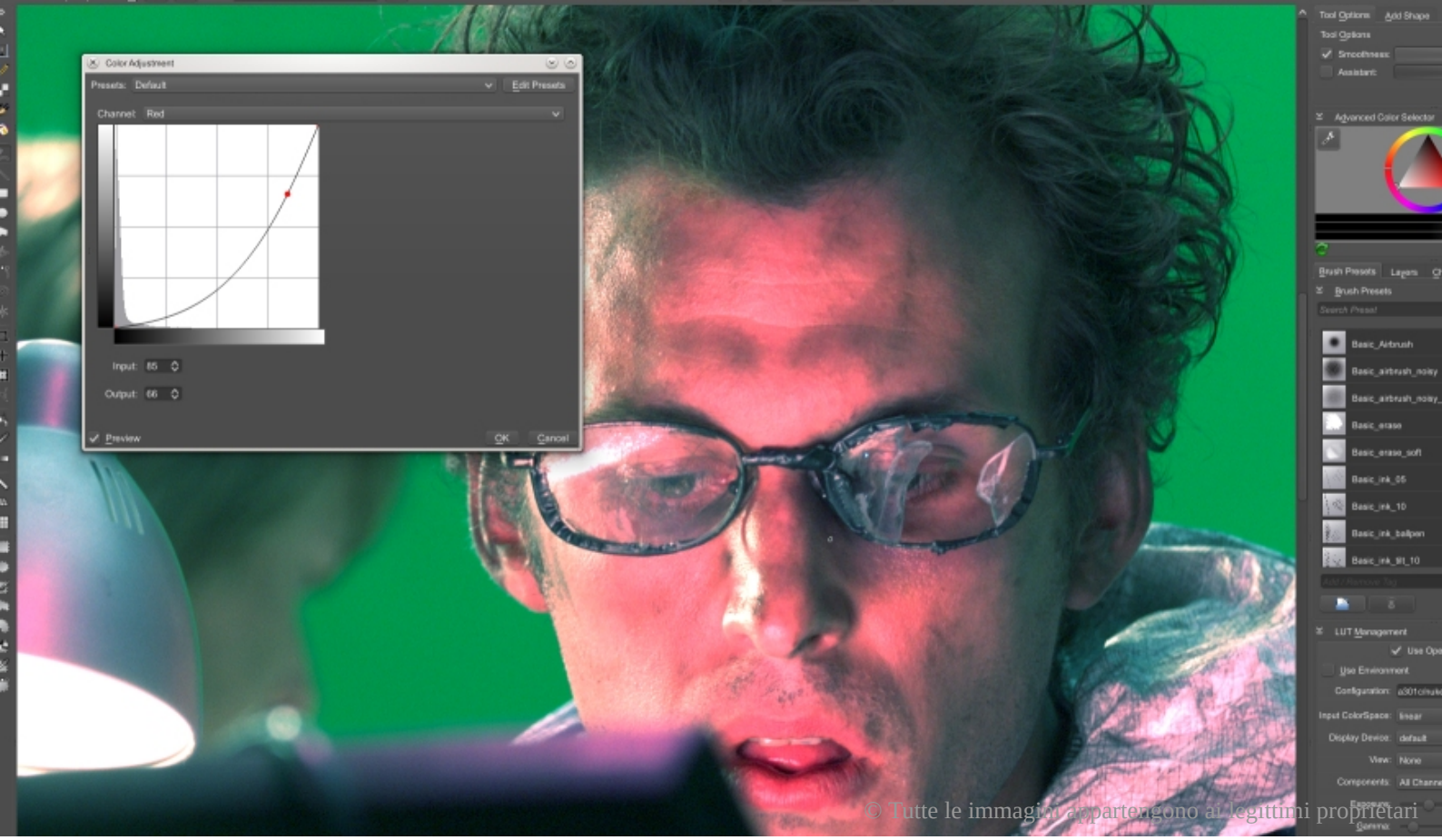
Un discorso che può sembrare inutile per il 90% di noi che lavora solo a piccoli progetti e molte volte prodotti personali, ma la visibilità è tutto nella vita di un grafico.
Si dice che Leonardo da Vinci abbia reso famosa la Gioconda dopo che il ritratto fu rifiutato dal committente, semplicemente portandosela dietro ad ogni occasione importante.
Al giorno d'oggi abbiamo altri mezzi per renderci visibili ad un pubblico più vasto possibile.
Esistono i social network, di cui voglio sfatare il mito della violazione della privacy, con i dovuti accorgimenti possono diventare dei canali molto importanti per farsi conoscere.
Non dimentichiamo i mezzi di informazione come riviste, forum e portali.

A volte basta poco per avere un po' di successo, ad esempio potreste creare un video o un'immagine che riguardi la vostra città e mandarla ad un quotidiano locale.

Come vedete, esistono diversi molti modi per rendersi visibili, quindi non resta che rimboccarci le maniche e farci conosce al mondo intero.

Alfonso Annarumma (anfeo)

In copertina:
"Natale a Gotham City"
di Seregost



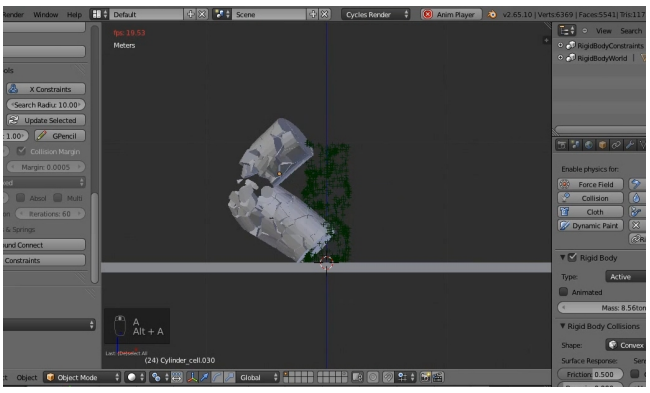
Blender: le news

Ultime news dal mondo Blender e non solo

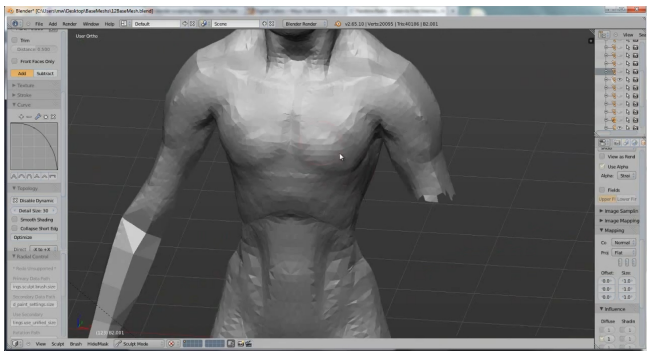
di Luca Pinciani

Blender 2.66

Da pochi giorni è uscita l'attesa nuova release di Blender, la numero 2.66 con tante novità.
La libreria Bullet per la fisica è stata integrata nel sistema di editor e animazione. Questo permette un workflow più semplice e veloce quando occorrono simulazioni fisiche. Un video che mostra in modo efficace questa novità lo potete trovare all'indirizzo
www.youtube.com/watch?v=xZV3M00ZzIU



Un'altra importante modifica riguarda Cycles: è stato infatti aggiunto il supporto per il rendering per peli e capelli.
Per la modellazione, un forte miglioramento è stato fatto per lo sculpt. Infatti è stata introdotta una nuova modalità di scultura che consente di suddividere la mesh solo dove necessario, molto simile alla DynaMesh di Zbrush. Un video che mostra le potenzialità di questa modalità lo potete vedere su Youtube:
www.youtube.com/watch?v=l3DLTnMEz14



Infine, ovviamente, molte altre sono le novità di Blender 2.66. Per leggerle tutte, potete consultare

il release log sul sito ufficiale http://www.blender.org/development/release-logs/blender-266/

2012

Anche se il 2012 è finito da tempo, diversi sono stati i bilanci e gli elenchi fatti relativi a Blender.
Ad esempio, Andrew Price di BlenderGuru.com ha stilato una personale classifica delle venti immagini più belle realizzate con Blender nel 2012. Se ve la siete persa, la trovate qui: http://www.blenderguru.com/top-20-blender-artworks-from-2012/

Il prestigioso sito CGSociety, invece, come ogni anno, ha selezionato i fatti più importanti legati alla computer grafica dell'anno appena trascorso ed al quinto posto è stato inserito Blender 2.65. Per leggere anche tutte le altre posizioni, il link è http://www.cgsociety.org/index.php/CGSFeatures/CGSFeatureSpecial/cg_retrospective_2012
Infine, un'altra classifica interessante è quella intitolata "Le migliori pubblicità realizzate con Blender nel 2012". La potete trovare alla pagina http://libregraphicsworld.org/blog/entry/best-commercials-created-with-blender-in-2012 nella quale vi sono incorporati i sette video scelti.

Non solo Blender: nuove release

In questi ultimi mesi anche diversi software legati al mondo Blender hanno ricevuto aggiornamenti.

MakeHuman (www.makehuman.org), il tool opensource per la creazione di personaggi 3d, è arrivato alla versione 1 Alpha 7 con molti miglioramenti e l'aggiunta di nuovi script e library.

Anche un altro software opensource è stato recentemente aggiornato: è stata infatti rilasciata la versione 2.6 di Krita (www.krita.org), con centinaia di bug fix, miglioramenti alle prestazioni e sopratutto con l'integrazione di OpenColorIO (www.opencolorio.org).
Infine, recentemente, anche due motori di render sono stati aggiornati: Indigo e LuxRender.
Del primo, con l'attuale versione 3.4 sono state notevolmente migliorate le prestazioni. Anche l'exporter di Blender (Blendigo) è stato aggiornato per la versione 3.4 di Indigo.
La nuova versione di LuxRender, la 1.2, invece, corregge numerosi bug: per i dettagli si può consultare il sito www.luxrender.net/forum/viewtopic.php?f=12&t=9653

Making of "Scarecrow"

L'immagine vincitrice del contest di Blendercoockies

di Alfonso Annarumma (anfeo)

L'immagine è stata realizzata per il contest di Halloween 2012 indetto da Blendercoockie dal tema “Scarecrow”.

Concept

Devo ammettere di non aver pensato molto a possibili varianti, l'idea era quella di realizzare un classico spaventapasseri, ma da tocco horror, quindi la prima cosa che ho fatto è cercare qualche immagine su google scrivendo semplicemente Scarecrow nel motore di ricerca. Se provate a fare lo stesso salteranno fuori forse le stesse immagini che mi hanno ispirato, una su tutte è stata quella in cui si vede una statua che raffigura una testa incappucciata e sotto si vede un occhio forse di uno zombie o altro mostro.

Subito prendo carta e matita e inizio a disegnare uno schizzo veloce (fig1). Il punto focale doveva essere l'occhio spalancato, interrogativo e spaventato.

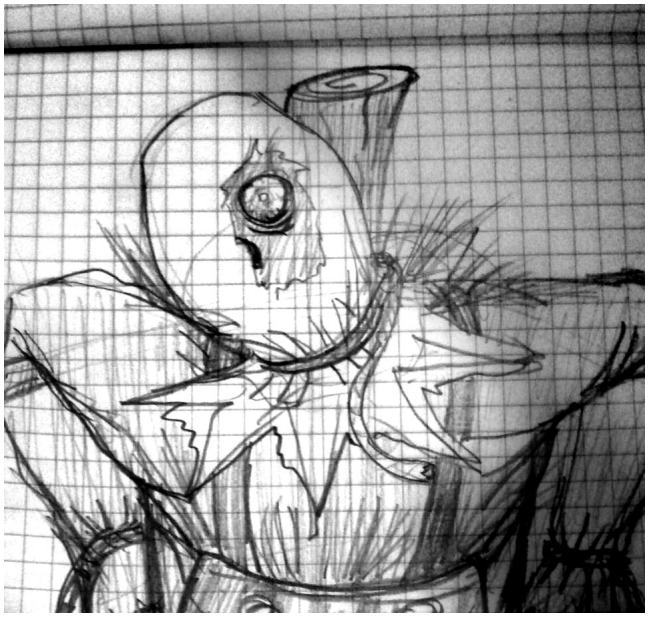


fig.1

Modellazione

Con lo schizzo sotto mano ho iniziato a modellare le forme di base (fig2). In questa fase non ho usato lo schizzo come ricalco, ma ho modellato tutto ad occhio, soprattutto per avere più libertà in seguito nella fase di sculpt (fig3).

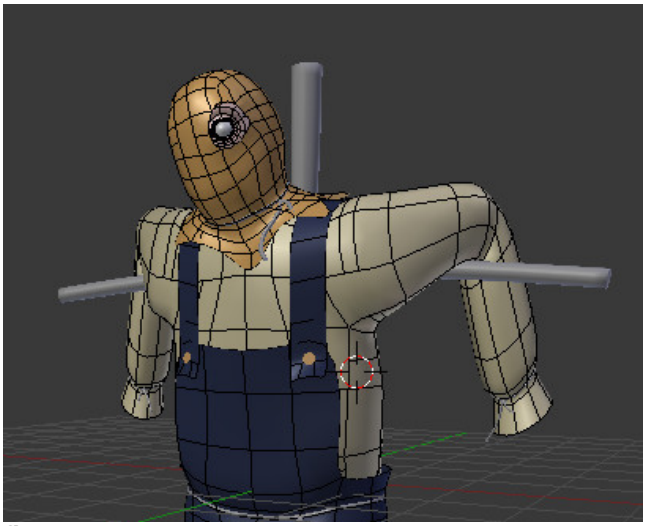


fig.2


fig.3

Alla fine ho mantenuto le mesh ad alta risoluzione, anche perché doveva essere solo una still quindi la quantità di poligoni era relativa (fig4).

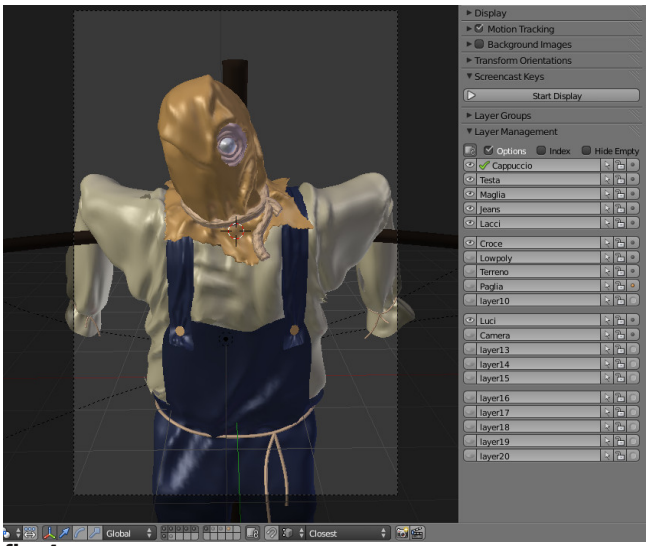


fig.4

La corda al collo non è altro che un semplice cilindro poi unwrappato (fig5) e ho usato il displacement per ottenere il risultato finale (fig6).

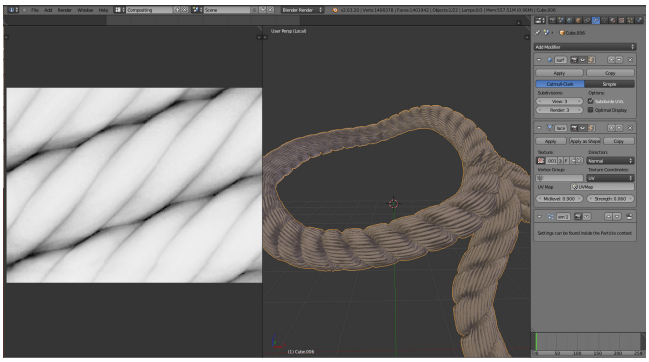


fig.5


fig.6

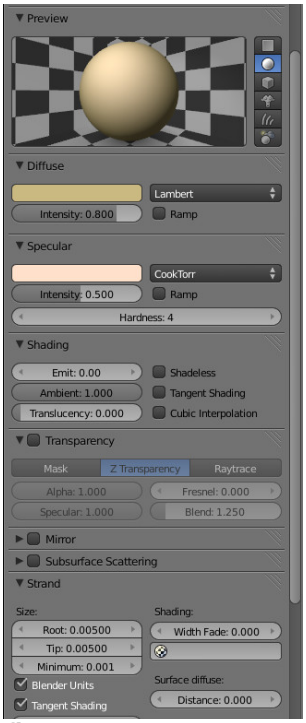


fig.7

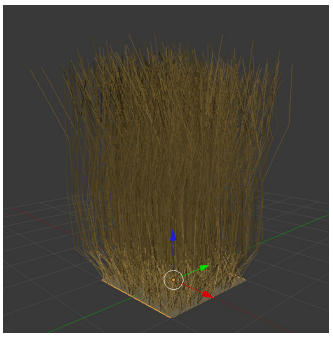


fig.8

Paglia

Per realizzare la paglia ho fatto prima qualche test in un file diverso, usando gli hair particles system di Blender (fig7). Il trucco per ottenere uno spessore corretto è quello di settare correttamente la dimensione degli Strand nelle opzioni del materiale (fig8).

Granturco

Il granturco l'ho realizzato mediante modellazione poligonale e project paint per disegnare le venature sulle pannocchie e le foglie

fig.9

utilizzando delle foto di riferimento prese da google e cgtextures (fig9).
Usando poi il particles system di Blender ho creato il campo di granturco usando un semplice piano.

Materiali e textures:

Dopo aver unwrappato tutto ho realizzato le varie textures. Sono partito utilizzando dalle texture seamless per realizzare le varie trame dei tessuti (fig10). Usando queste come base ho realizzato dei bake di Normal Map, Color map e AO.
Le texture finali le ho poi dipinte in Gimp usando Color e AO per definire i colori iniziali e aggiungendo vari livelli di sporco e smagliature varie. (fig11).

fig.10

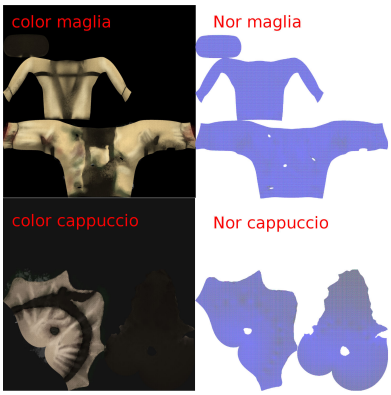


fig.11

Per la pelle ho usato il classico sistema a nodi con la fusione di diversi livelli di SSS (fig16a):
- backscatter
- subdermal
- epidermal
- hard spec
- soft spec

Pelucchi

Per realizzare i pelucchi di juta, maglia e corda ho usato di nuovo il particle system (fig12). Il trucco è di giocare un po' con i settaggi dei children per ottenere l'effetto scompigliato. Poi basta settare correttamente il materiale usando una texture blend su alpha per rendere le punte dei peli sfumate.

fig.12

Luci

L'idea per l'illuminazione era quella di avere una qualche luce lunare in alto a sinistra e di fronte una luce calda di contrasto. L'effetto finale l'ho ottenuto dopo numerose prove e soprattutto grazie ai consigli che mi sono arrivati tramite i vari forum su cui ho proposto il wip.
La luce lunare che arriva da in alto a sinistra parte da una Spot Lamp leggermente azzurra molto soft e con ombra molto sfumata. La si può vedere molto sul cappuccio dello spaventapasseri.
Di rinforzo a questa Spot ho messo un'altra Spot Lamp alle spalle in modo che simulasse una classica Rim light. Quest'ultima, anche se non si vede molto è di colore verde, sempre molto

chiaro.
La luce che invece dà l'atmosfera vera e propria è una semplice Lamp con spuntata la casella Sphere, che ne limita il falloff ad una distanza preimpostata. Delle luci usate quest'ultima è l'unica che non proietta le ombre (fig.13).

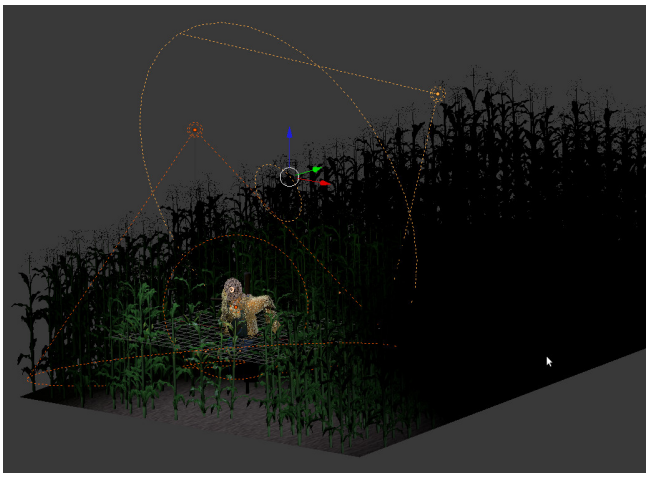


fig.13

L'AO impostato come multiply definisce il resto delle occlusioni e ombre che danno profondità ai dettagli modellati.

Rendering e Postproduzione

L'immagine è stata renderizzata ad una risoluzione di 1200x1600. Il formato verticale è quello che preferisco perché mi ricorda sempre quello delle locandine dei film.
La Postproduzione l'ho fatta in Blender con l'ottimo node editor. Per prima cosa ho diviso il render in 3 render layer, Campo, Scarecrow, Primo Piano (fig14).

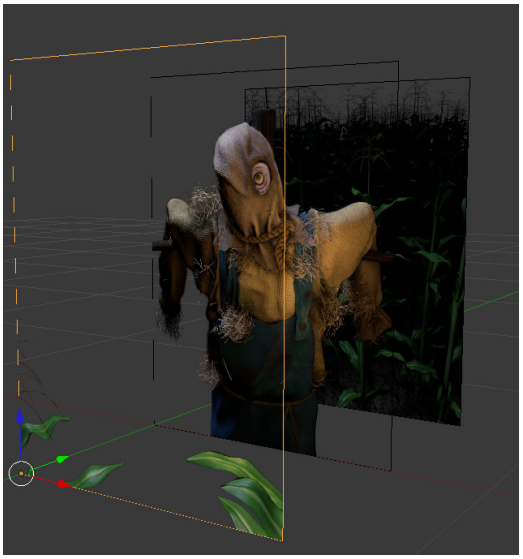


fig.14

Dai nodi (fig15) si possono intuire i vari passaggi:

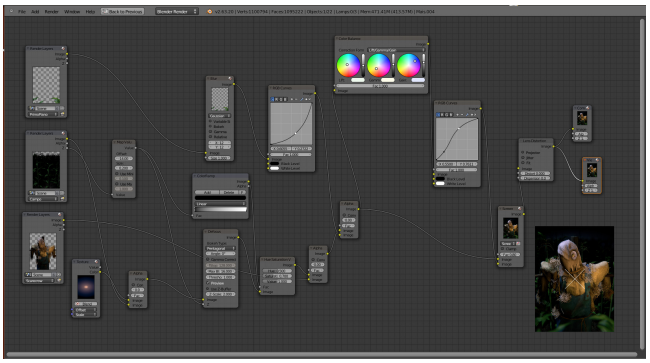
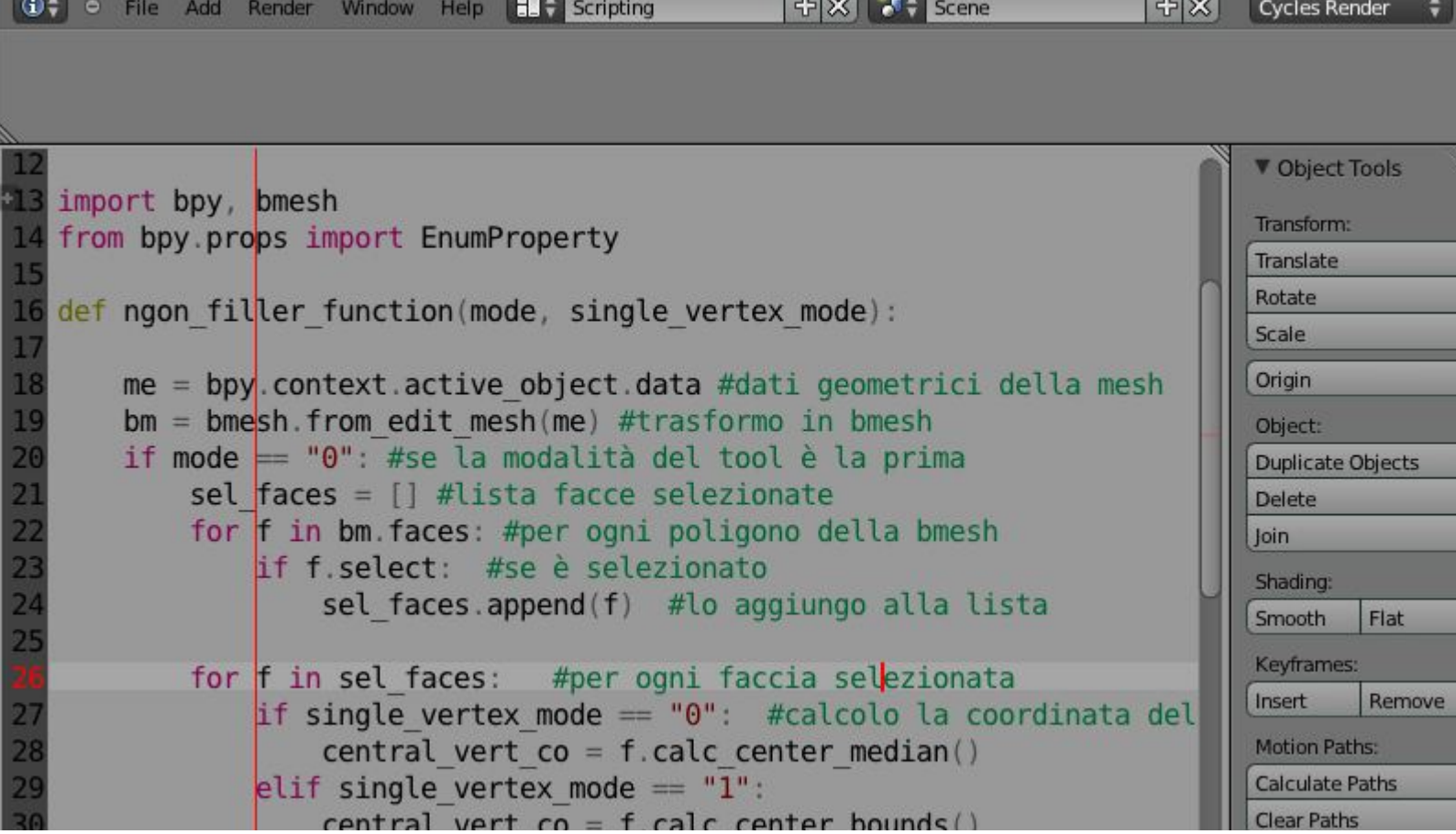


fig.15

ricavo una Zdepth dal livello Campo, poi aggiungo un cielo con una Texture Blend
applico il defocus al Campo e desaturo usando la Zdepth come riferimento per ottenere maggiore profondità, così ottengo lo sfondo per l'immagine
sfoco le foglie in Primo Piano e infine fondo i tre livelli
color correction per un look più horror con ombre sul verde e luci sul blu
un ritocco di Curve RGB per migliorare i contrasti
un briciolo di color aberration e la prima parte di post produzione è finita.

In The Gimp ho migliorato l'immagine con una maschera di contrasto e ho schiarito la parte centrale dell'immagine per accentuare la presenza della luce che illumina il volto.



Tutorial: script in Python

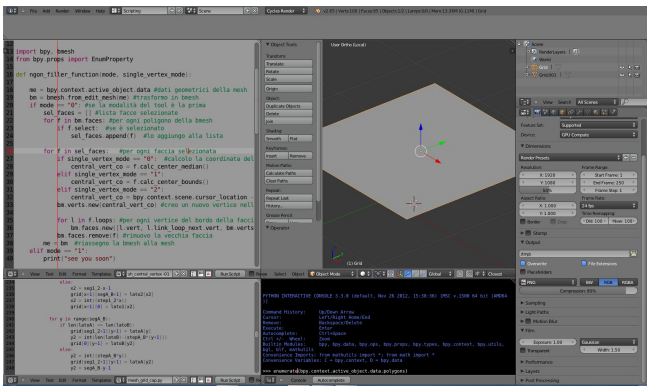
Impariamo il Python con un semplice script

di Emanuele Bernardi

Questo tutorial riguarda la creazione di un piccolo script in python per Blender, e ha come obbiettivo lo spiegare in maniera semplice qualche concetto base sia del linguaggio Python che del modo di trattare i dati interni di Blender per ottenere nuove funzionalità. Io non sono un programmatore professionista, ho sempre approcciato la programmazione in modo amatoriale e pratico, quindi le spiegazioni non saranno sicuramente perfette dal punto di vista teorico, ma più che altro avranno un significato pratico. Quindi non perdiamoci in ulteriori chiacchiere e passiamo al sodo.

Lo script che analizzerò è uno a cui sto lavorando, e permette di sostituire una faccia già esistente con un vertice centrale e una serie di facce triangolari che lo connettono al bordo della faccia precedente. Diciamo che non è di grande utilità, nel senso che lo stesso risultato si può ottenere estrudendo e scalando a zero la nuova faccia, ma comunque permette di risparmiare qualche passaggio. Da considerare anche che permette due diverse modalità di creazione del vertice che vedremo in seguito.

Per iniziare a creare uno script impostiamo il layout di Blender in modalità scripting. Lo fa in automatico scegliendo il layout dal menù a tendina in alto, dove di default c'è scritto appunto Default. Dovrebbe comparire simile a questo. Io l'ho modificato un po' dando più spazio alla zona di testo rispetto alla Console Python (quella nera con le scritte blu).



Fatto questo, dobbiamo creare un nuovo documento di testo premendo il pulsantone New in fondo al text editor. Qui possiamo cambiare il nome al nuovo testo con qualcosa di più rappresentativo, tipo “crea_vertice_centrale”. Ora è possibile scrivere nell’editor. Quello che abbiamo creato è un file di testo che è presente solo nel file .blend su cui stiamo lavorando. Quindi quando salveremo il .blend, il file di testo si salverà in esso.

I programmatori esperti preferiscono scrivere il codice in altri editor di testo, tipo Notepad++, perchè hanno funzioni migliori per l’editing del codice. Questo però comporta il dover salvare ogni volta il file, ritornare in blender, aggiornare il file nell’editor e far andare lo script. Io però, non essendo esperto, ho bisogno di fare molti test per vedere se lo script funziona, per cui questo metodo mi rallenterebbe molto. Per questo preferisco lavorare direttamente nel text editor di Blender, in modo che con un click posso testare il mio script.

E’ ora utile attivare due funzionalità dell’editor di testo. Vicino al pulsante Run script in basso, ci sono tre pulsantini. Attivare il primo, per avere il numero delle linee vicino al codice, e attivare il terzo per avere le parole chiave del codice colorate. Questo è utile per orientarsi meglio nel codice. Il secondo pulsantino fa andare a capo le linee di codice che sono troppo lunghe per la finestra del codice. Io preferisco non attivarlo perchè ho paura di fare confusione credendo che una linea sia finita quando non lo è.

Ora iniziamo finalmente a scrivere qualche riga, e successivamente vedremo alte utili funzioni dell’editor di testo. Per scrivere qualunque script per Blender bisogna innanzitutto dirgli quali componenti andremo ad utilizzare. I componenti sono contenuti in dei moduli. La maggior parte dei componenti sono contenuti nel modulo bpy (BlenderPYthon). Quindi la prima riga sarà:

```
import bpy
```

questo quindi dice al programma, in parole povere, di tenersi pronti tutti i componenti di quel modulo, in modo che quando richiamati siano disponibili.

Ora aggiungiamo una linea per fare un primo test (scrivete tutto, compresa la parte dopo il cancelletto):

```
print(“Hello world!”)     #questa linea scrive Hello world nella console
```

Adesso dopo aver scritto questa linea clickate sul pulsante Run script in fondo all’editor. Non dovrebbe succedere nulla, o almeno sembra. Per vedere se è successo qualcosa dobbiamo aprire la System Console, andando sul menu in alto Window e scegliere Toggle System Console. Comparirà una finestra nera con delle scritte bianche. Quando si usa il comando print Blender scrive qui quello che gli indichiamo. Dovreste trovarci scritto Hello world.

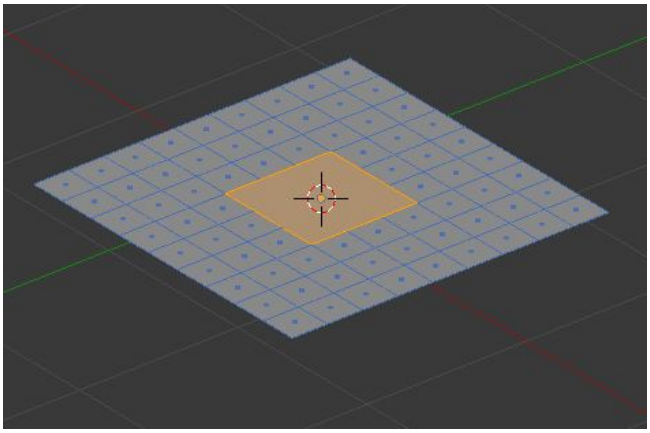
Se dovesse dare qualche errore, mi raccomando di controllare bene che sia scritto tutto come indicato, rispettando minuscole, maiuscole, spazi, punteggiature ecc. In generale in Python si scrive sempre in minuscolo, con alcune eccezioni.

Avrete notato che dopo il comando ho messo un carattere cancelletto (#) e scritto delle cose. Quello è un commento. Tutto ciò che viene scritto a destra del carattere cancelletto viene ignorato dal programma, e serve solo al programmatore per scrivere cosa fanno le righe di codice e il motivo di certi comandi piuttosto che di altri. E’ molto importante scrivere i commenti, perché a volte certe sequenze di comandi non sembrano avere senso, ma quando le si scrisse lo si fece per qualche motivo. Meglio quindi lasciare un promemoria nel caso si riprendesse in mano il codice molto tempo dopo o per altri programmatori.

Ora potete cancellare la riga col comando print. Questo comando tornerà utile quando vorremo

verificare i progressi del nostro script.

Passiamo alle righe necessarie per il nostro scopo. La prima cosa da fare ora è recuperare i dati della mesh su cui vogliamo lavorare. Per fare i nostri test avremo anche bisogno di una vera mesh. Quindi creiamo una griglia come in figura, unendo le nove facce centrali in un n-gon usando il tasto F. Dobbiamo rimanere in edit mode e con quella faccia selezionata, perchè sarà in questa modalità che lo script dovrà lavorare.



Ora nell'editor scriviamo:

```
me =
bpy.context.active_object.data
#dati geometrici della mesh
bm = bmesh.from_edit_mesh(me)
#trasformo in bmesh
```

Nella prima riga andiamo ad assegnare tutta quella pappardella ad una parola molto più breve. La parola me è una variabile, con un nome che possiamo decidere noi. Da lì in avanti sapremo che me contiene bpy.context.active_object.data, e scrivere me sarà come scrivere bpy.context.active_object.data.
In questo caso "me" sta per "mesh", e bpy.context.active_object.data sono proprio i dati della mesh.

Andare a leggere i dati della mesh è come navigare in delle cartelle. La cartella madre è bpy, la quale contiene delle sottocartelle, e una di queste è context. Dentro context c'è active_object, che è l'oggetto attivo in quel momento, e dentro l'oggetto attivo ci sono i dati della mesh, oltre a tutti gli altri dati e i materiali ecc. Dentro "data" a loro volta ci saranno vertici, facce ecc che andremo a vedere successivamente.
Il segno uguale serve per eseguire l'assegnazione della parte che sta a destra nella parte che sta a sinistra.

Ora, per lavorare sfruttando le funzionalità delle nuove bmesh, dobbiamo convertire i dati mesh in dati bmesh. Ed è quello che fa la seconda riga. Inserisce nella variabile bm (che starebbe per bmesh) la bmesh creata dalla funzione from_edit_mesh. Fra le parentesi della funzione inseriamo i dati mesh precedentemente creati, ovvero la variabile me, che sarebbe la mesh da convertire in bmesh.

Provate ora a fare Run Script dopo aver scritto queste ultime due righe. Dovrebbe darvi un segnale di errore. Per vedere di che errore si tratta bisogna andare nella System Console che abbiamo aperto prima.
Dopo Hello world! Che avevamo scritto prima, dovrebbe esserci scritto

```
Traceback (most recent call last):
  File "\Text", line 4, in <module>
NameError: name 'bmesh' is not defined
Errore: Python script fail, look in the console for now...
```

Se guardiamo bene, le informazioni che ci interessano sono dove è l'errore, e quale è. L'errore si trova alla linea 4 nel mio caso, visto che ho lasciato una riga vuota fra la prima e le altre due righe. In ogni caso dovrete andare alla linea indicata dell'editor. L'errore è "name bmesh is noto defined", ovvero il programma non sa cosa sia "bmesh". Questo perchè la funzione from_edit_mesh non appartiene al modulo bpy, ma ad un modulo che non abbiamo importato, ovvero il modulo "bmesh". Quindi per far funzionare lo script fino a qui dobbiamo importare anche il modulo bmesh, aggiungendolo nella prima riga in questo modo:

```
import bpy, bmesh #separare i
moduli importati con una virgola
```

Dopo questa modifica non dovreste più riscontrare l'errore, anche se in effetti lo script non esegue nessuna azione.

Ora che abbiamo dei dati su cui lavorare, dobbiamo trovare le facce su cui eseguire la conversione, ovvero le facce che l'utente ha selezionato. Bisogna passare in rassegna tutte le facce e vedere quali sono selezionate, e poi inserirle in una lista.

Per prima cosa creiamo la lista vuota scrivendo:

```
sel_faces = []    #le parentesi quadre rappresentano una lista vuota
```

ora eseguiremo un ciclo di controllo sulle facce per vedere se sono selezionate, e in caso affermativo le aggiungiamo alla lista.

```
for f in bm.faces: #per ogni poligono della bmesh
if f.select:  #se è selezionato
sel_faces.append(f) #lo aggiungo alla lista
```

bm.faces è la lista di tutte le facce della bmesh che abbiamo creato prima. Grazie al comando For, il programma assegnerà di volta in volta una faccia della bmesh alla variabile f. Usare i due punti alla fine della riga per indicare che la riga del ciclo For è conclusa.
Fatto questo, nella riga successiva si creerà un indentamento, ovvero degli spazi vuoti rispetto all'inizio della riga precedente. Si ottiene con il tasto tab nel caso non venisse in automatico.
Grazie al comando if vado a vedere se la proprietà della faccia f.select (che indica se la faccia è selezionata) è vera. Avrei anche potuto scrivere:

```
if f.select == True:    #controlla se la proprietà è vera
```

notare che per confrontare l'uguaglianza fra due proprietà si usano due segni di uguale (==), mentre un segno di uguale (=) serve per assegnare un valore ad una variabile o proprietà. Le parole chiave True e False vanno scritte con la maiuscola. La riga di confronto va chiusa con i due punti.
Se non scrivo == True, il programma dà per scontato che chieda se la proprietà è vera.

Ora dovrebbe crearsi un secondo indentamento, per indicare che le prossime istruzioni vanno eseguite solo se tutte le condizioni espresse nella riga dell'if vengono soddisfatte. In questo caso solo se la proprietà f.select è vera. La terza riga usa il comando append per aggiungere quella faccia alla lista. Il comando append va usato scrivendo la lista alla quale si vuole aggiungere un elemento, poi il .append, e poi fra parentesi tonde l'elemento da aggiungere. In questo caso la faccia f.

Ora per verificare che il codice faccia quello che vogliamo, scriviamo nella riga dopo

```
print(sel_faces)
```

questo comando deve essere allineato con la riga del ciclo for, perchè deve essere eseguito dopo che tutte le facce della bmesh sono state controllate. Ora nella viewport proviamo a selezionare un numero conosciuto di facce, tipo 4, e poi far partire lo script con Run script. Nella console ora dovrebbe essere apparsa una lista di numeri un po' enigmatici. Inizia con una parentesi quadra, e quindi è una lista di cose, e in questo caso la lista delle facce che sono finite nella nostra lista. Poi ci dev'essere scritto <BMFace e un serie di dati, fra cui un index. Questi sono i dati della faccia in lista. Dovrebbero essereci tante BMFace quante sono le facce della griglia che avete selezionato. Se è così allora fino a qui il codice funziona!

Ecco il codice scritto fin qui

```
import bpy, bmesh

me =
```

```
bpy.context.active_object.data
#dati geometrici della mesh
bm = bmesh.from_edit_mesh(me)
#trasformo in bmesh

sel_faces = []
for f in bm.faces: #per ogni
poligono della bmesh
    if f.select:  #se è
selezionato
        sel_faces.append(f)  #lo
aggiungo alla lista
print(sel_faces)
```

Procediamo! Cancellate pura la riga print. Ora che abbiamo la lista delle facce selezionate, possiamo procedere alla conversione di ciascuna. Useremo un ciclo for per prendere ogni faccia, troveremo le coordinate del punto centrale dove creare il vertice, poi creeremo delle facce triangolari fra il vertice creato e quelli del bordo, e poi cancelleremo la vecchia faccia.

Il ciclo for viene iniziato così:

```
for f in sel_faces:   #per ogni
faccia selezionata
```

Ora possiamo usare due varianti per la coordinata del vertice centrale: o posizionarlo sul punto medio di tutti i punti del bordo della faccia, o posizionarlo sul centro geometrico dell'ingombro della faccia. Per ciascuno dei due metodi c'è una funzione apposita.

Sulla riga successiva facciamo un'indentatura e scriviamo:

```
if single_vertex_mode == "0":
#calcolo la coordinata del nuovo
centro
                central_vert_co =
f.calc_center_median()
elif single_vertex_mode == "1":
                central_vert_co =
f.calc_center_bounds()
```

Qui abbiamo fatto una scelta fra due possibilità. Nella prima riga abbiamo usato l'if come sopra, e nel caso quella condizione sia vera, il programma eseguirà la seconda riga. Se però la prima condizione non dovesse risultare vera, allora diamo una seconda condizione, e lo facciamo usando elseif con una seconda condizione. Potremmo andare avanti ancora inserendo nuove condizioni e azioni da fare. Con questo metodo, quando trova una condizone vera, non va a guardare le successive.

In queste righe di codice abbiamo controllato il valore di una variabile, ma ancora questa variabile non è stata creata. Quindi dobbiamo aggiungere una riga prima di iniziare il ciclo for, dove definiamo il valore di questa variabile.

```
single_vertex_mode = "0" #da
inserire prima del ciclo for
```

Dopo ciascuna riga di controllo c'è un'indentatura dove diciamo al programma cosa fare nel caso quella condizione sia vera. Nel primo caso diciamo di assegnare il risultato di una funzione ad una variabile. La funzione f.calc_center_median() calcola il punto medio delle coordinate di tutti i vertici del bordo della faccia f, che è una di quelle nella nostra lista di facce selezionate. Nella quarta riga la funzione calcola il centro geometrico della faccia e la assegna alla stessa variabile.

Riassumendo, in base al valore che diamo noi alla variabile single_vertex_mode, il programma assegnerà alla variabile central_vert_co le coordinate del punto medio o del punto centrale della faccia.

Quando abbiamo ottenuto le coordinate, possiamo creare un vertice a quelle coordinate. Scriviamo:

```
bm.verts.new(central_vert_co)
#creo un nuovo vertice nella
coodinata trovata
```

Questa funzione va nella lista dei vertici della bmesh e dice di crearne uno nuovo, e fra parentesi abbiamo messo la variabile con le coordinate precedentemente trovate, in modo che per ciascuna faccia il programma crei il vertice al posto giusto.

Il codice visto quindi in questa fase è il seguente:

```
single_vertex_mode = “0” #imposto
la modalità di creazione del
vertice

for f in sel_faces:   #per ogni
faccia selezionata
if single_vertex_mode ==
"0":#calcolo la coordinata del
nuovo centro
               central_vert_co =
f.calc_center_median()
          elif single_vertex_mode
== "1":
          central_vert_co =
f.calc_center_bounds()

     bm.verts.new(central_vert_co
)#creo un nuovo vert nella cood.
trovata
```

Per poter fare una prova del funzionamento del codice fin qui dobbiamo aggiungere una riga di codice.
Fuori dal ciclo for, quindi allineata con la parola for, scrivete:

```
bmesh.update_edit_mesh(me)#aggiorn
o la mesh coi dati della bmesh
```

In questo modo la nostra mesh verrà aggiornata con i nuovi vertici creati. Scritto questo potete prendere la vostra griglia, selezionare alcune facce, e premere Run Script. Dovrebbero apparire dei vertici al centro delle facce selezionate (li potete vedere solo in modalità vertici ovviamente).

Proseguiamo! Ora dobbiamo creare le nuove facce. Siccome questo viene fatto per ciascuna faccia selezionata, continuiamo a scrivere il codice dentro al ciclo for, in modo che lo esegua per ogni faccia selezionata.
Qui diremo al programma che per ogni vertice del contorno della mesh, deve creare una faccia che ha come vertici il vertice che stiamo guardando, il successivo, e il vertice creato prima. Per fare questo useremo un nuovo ciclo for all’interno del primo. Per passare in rassegna tutti i vertici del bordo di una faccia dobbiamo usare i loop, che non sono la stessa cosa di quelli formati dagli edge nella modellazione.
Il ciclo for inizia così:

```
for l in f.loops:   #per ciascun
loop della faccia f
```

ora per creare la nuova faccia dobbiamo usare la funzione bm.faces.new()
Fra parentesi devo inserire i 3 vertici che la definiscono. Partiamo dal vertice del loop, che si indica con l.vert. Il vertice successivo invece si trova indicando il loop successivo e poi il suo vertice, quindi grazie alla funzione link_loop_next sarà l.link_loop_next.vert. Il vertice che abbiamo creato invece è l’ultimo della lista. In Python, per prendere l’ultimo elemento di una lista si conta all’indietro, quindi sarà
bm.verts[-1]. Se avessi voluto prendere il penultimo avrei scritto -2 e così via.

Quindi la funzione di creazione della faccia risulterà (inserita nel ciclo for):

```
bm.faces.new([l.vert,
l.link_loop_next.vert,   bm.verts[-
1]])
```

Attenzione! I 3 vertici vanno scritti fra parentesi quadre, oltre che fra le tonde della funzione, probabilmente perchè è una lista di elementi.

Ora non ci resta che rimuovere la vecchia faccia, scrivendo semplicemente:

```
bm.faces.remove(f)
```

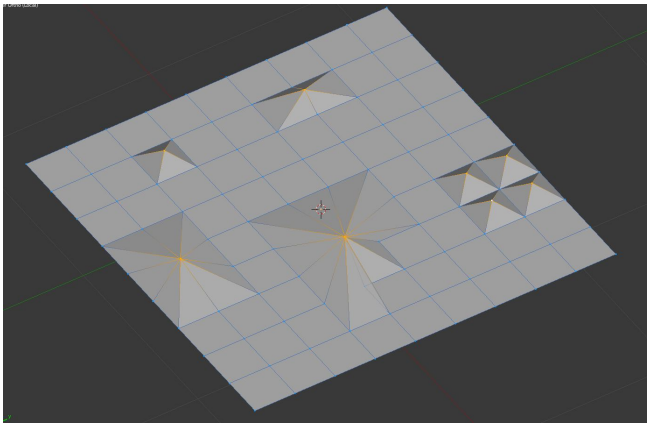
Questo va scritto all'esterno del ciclo for dei loop, ma all'interno del ciclo for delle facce!
Riassumendo la fase di creazione delle geometrie:

Dopo aver creato la lista di facce selezionate, imposto la modalità. Per ciascuna faccia, calcolo la coordinata del centro in basse alla modalità. Creo il vertice alla coordinata trovata. Poi per ciascun vertice del bordo della faccia, creo una faccia fra due vertici consecutivi del bordo e il vertice centrale. Poi cancello la faccia e riassegno la bmesh con le nuove geometrie alla mesh dell'oggetto.

```
import bpy, bmesh

me =
bpy.context.active_object.data
#dati geometrici della mesh
bm = bmesh.from_edit_mesh(me)
#trasformo in bmesh

sel_faces = []
for f in bm.faces: #per ogni
poligono della bmesh
    if f.select:  #se è
selezionato
        sel_faces.append(f)  #lo
aggiungo alla lista

single_vertex_mode = “0” #imposto
la modalità di creazione del
vertice

for f in sel_faces:   #per ogni
faccia selezionata
if single_vertex_mode ==
"0":#calcolo la coordinata del
nuovo centro
               central_vert_co =
f.calc_center_median()
          elif single_vertex_mode
== "1":
          central_vert_co =
f.calc_center_bounds()

     bm.verts.new(central_vert_co
)#creo un nuovo vert nella cood.
trovata

     for l in f.loops:  #per
ciascun loop della faccia f
     bm.faces.new([l.vert,
l.link_loop_next.vert, bm.verts[-
1]])

     bm.faces.remove(f)#rimuovo
la faccia vecchia
```

bmesh.update_edit_mesh(me) #riassegno la bmesh alla mesh

E questo può essere il risultato dello script con una serie di facce di varie forme. Il vantaggio rispetto che farlo manualmente è che si può fare su molte facce diverse in un colpo solo.



Nel prossimo numero vedremo come creare un add-on per questo script, in modo che sia più facile da eseguire, e magari qualche altra funzione.
Se avete problemi o dubbi o richieste di chiarimento scrivete pure sul forum di Blender Italia. Tanti saluti!

Tutorial: uva realistica

Creare un grappolo di uva realistico con Blender

di Bernardo Iraci

Buongiorno a tutti i lettori di BMI! In questo tutorial vedremo come realizzare un grappolo d'uva partendo da zero, texturizzarlo e creare i materiali. Cercheremo di raggiungere un risultato il più possibile realistico.
Per questo lavoro ho scelto di usare il blender internal, per via dell'SSS e della flessibilità che ancora offre in fatto di textures procedurali. Bando ai convenevoli e cominciamo!

MODELLAZIONE ACINO

Prima di cominciare vi consiglio di procurarvi delle buone references. Qualche bel grappolo e qualche close up sugli acini vi faranno buona compagnia durante la modellazione e la creazione dei materiali.

Per prima cosa creeremo un acino. Usiamo una UV sphere per cominciare. Per il numero di anelli e segmenti potete regolarvi in base al grado di dettaglio che volete. Per quanto mi riguarda dei valori rispettivamente di 10 e 20 hanno funzionato più o meno bene. Aggiustate la mesh

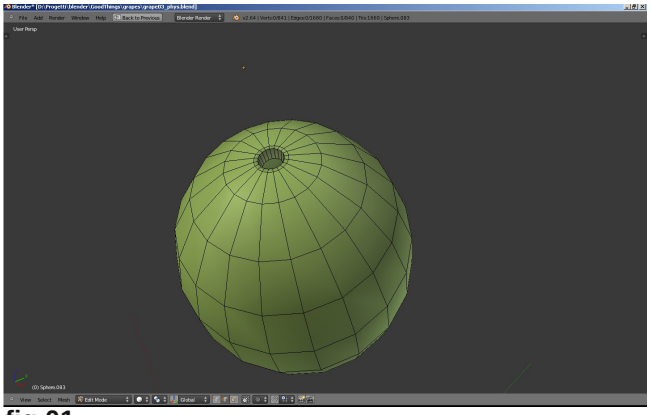


fig.01

scalandola verticalmente e aggiustandola a dovere per fargli assumere la forma tipica dell'acino. A questo punto dalla cima dell'acino create un incavo verso l'interno (fig.01) e quindi tornate fuori fino a creare il picciolo(FIG.02) (non so se ho usato il termine corretto, non me ne vogliano gli agrofili!).
Il picciolo, come vedete in figura, potete crearlo in maniera abbastanza "irregolare" per dargli un aspetto più organico. Tra breve comunque lo renderemo ancora più storto...

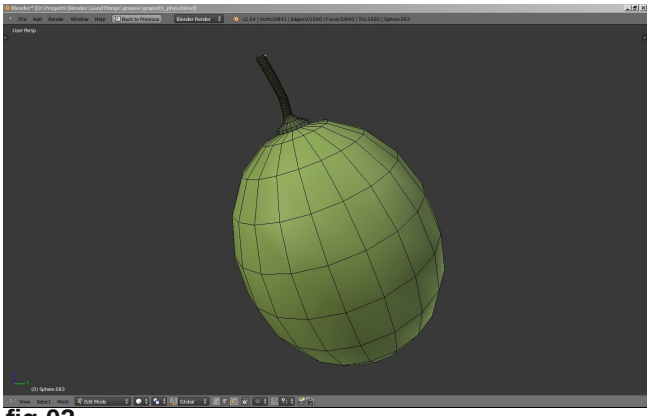


fig.02

Come potete notare nella figura 02 per il picciolo ho utilizzato un numero di edge loops abbastanza elevato. Questo perchè andremo ad applicarci ora un modificatore displace. Per le esigenze iniziali del mio progetto (che includeva una simulazione fisica del grappolo... ma di questo ne parleremo un'altra volta, se riuscirò nell'impresa) avevo scelto di non usare modificatori subsurf. Se voi preferite comunque, al posto dell'alto numero di edgeloop sul picciolo, potete usare un modificatore subsurf su tutta la mesh dell'acino.

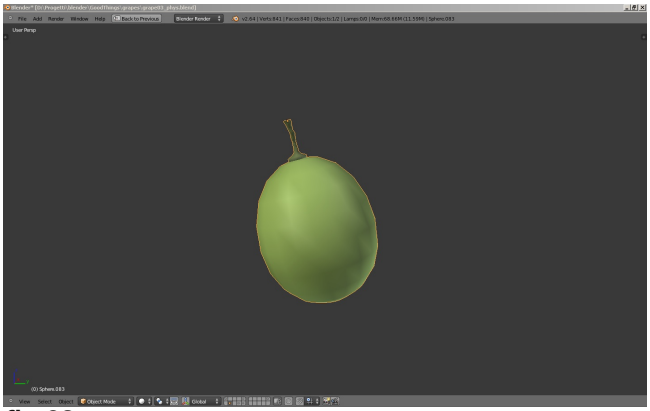


fig.03

Qualsiasi strada scegliate di intraprendere applichiamo ora questo displace alla mesh. Prima però creiamo un vertex group per il picciolo, in quanto non vogliamo che il displace venga applicato anche all'acino. Come texture una cloud andrà benissimo. Come tocco finale muovete leggermente qualche vertice dell'acino stesso aiutandovi con il proportional edit per dare un po' più di credibilità al tutto. Inoltre spostiamo il pivot all'apice del picciolo, cosa che ci aiuterà in seguito. Prima di concludere questa parte assegniamo un materiale alla mesh e chiamiamolo acino. Aggiungiamo un secondo materiale e assegniamolo al solo picciolo. Alla fine dovreste avere qualcosa di questo genere (fig.03)

UV E TEXTURES

Passiamo ora ad una rapida fase di unwrapping. Praticate un taglio lungo tutta l'altezza dell'acino e un altro in uno degli edge loop nascosti sotto la base del picciolo e quindi eseguite l'unwrap della mesh. (fig.04 e 05).

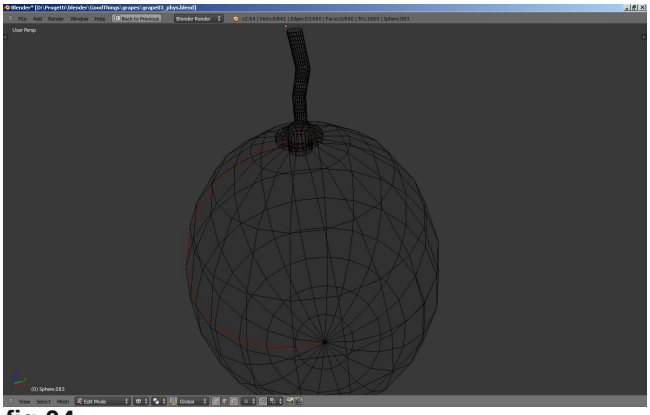


fig.04

Nell'image editor create una nuova texture (1k andrà benissimo) ed entriamo nella modalità di texture paint. Usando un pennello bianco andremo ora a dipingere la parte alta dell'acino. Se osservate delle reference infatti potrete notare che spesso nella parte più alta gli acini assumono un colore tendente al bianco. Fate si che la texture sia bella sfumata. Non preoccupatevi di esagerare con l'intensità del bianco... quando applicheremo la texture potremo regolarne l'intensità. L'importante è che non ci sia uno

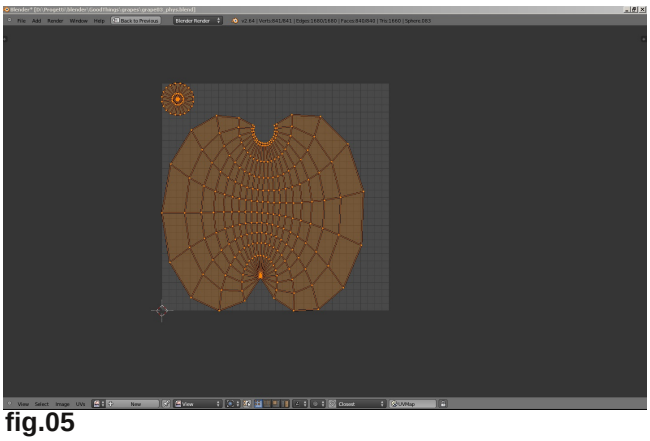


fig.05

stacco troppo netto. A questo scopo potete aiutarvi con lo smear ed il blur, o all'occorrenza ritoccare in seguito l'immagine in un programma esterno. Quando siete soddisfatti salvate la texture.

Create ora una seconda immagine sulla quale dipingeremo le venature tipiche dei chicchi d'uva. Ricordatevi sempre di tenere una reference sotto mano. Con il solito pennello bianco praticate delle linee verticali in modo irregolare... alcune più lunghe, altre meno. Alla fine date qualche mano leggera di smear per rendere il tutto più' organico. Di nuovo potete ritoccare in un programma esterno se preferite.

MODELLAZIONE RASPO

Qui possiamo intraprendere due strade diverse. In entrambi i casi comunque sarà necessaria una buona dose di olio di gomito.

Il primo metodo consiste nel distribuire prima gli acini usando una mesh come emitter di particelle e in seguito costruire in base alla posizione degli acini il raspo. C'è da notare che molto probabilmente un buon lavoro di aggiustamento della posizione dei singoli acini a posteriori sarà comunque necessario. In ogni caso le particelle ci aiuteranno a dare la forma iniziale. Se preferite comunque, potete fare anche tutto a mano.
Tuttavia il vedere la disposizione degli acini dall'inizio può darci una buona idea dell'aspetto che avrà il nostro grappolo a lavoro ultimato.

Nel secondo caso modellerete prima il raspo senza avere acini di mezzo e in seguito a mano posizionate un chicco per uno dove volete.
Io ho usato il primo metodo. Ed ecco come ho fatto.

Ho creato una Uvsphere e ho eliminato i vertici ai poli della sfera. A questo punto (ricordate di tenere sempre una reference sott'occhio) ho modificato la forma della sfera in modo da fargli assumere grossolanamente la forma di un grappolo d'uva.
Aggiungete quindi un hair particle system. I settaggi importanti sono i seguenti: emit from verts, rotation attivata e settata su normal, aumentare un po' il random size, disattivare la renderizzazione dell'emitter e ovviamente renderizzare le particelle come oggetti, usando la mesh del nostro acino.
Per quanto riguarda il numero di acini potete regolarvi come meglio credete. Calcolate che più acini avete e più sarà tedioso il lavoro che ci aspetta tra poco... (oltre ovviamente a darvi problemi di compenetrazione)

E' molto probabile che avrete bisogno di spostare ed eliminare diversi vertici della mesh usata come emitter. Potete anche aggiungere dei vertici all'interno della mesh principale. Insomma, guardate un grappolo reale e create una disposizione il più possibile realistica.
Una volta che siete soddisfatti della forma di massima del vostro grappolo dal menu dei modificatori convertite le particelle in mesh reali e spostate tutti i chicchi su un nuovo layer. (fig.06)
E' arrivato il momento di creare il raspo usando lo skin modifier. Create un piano, eliminate tre vertici, piazzate quello rimasto sul pivot e aggiungete il modificatore skin, più un subsurf. A questo punto è l'ora di sfoderare la vostra

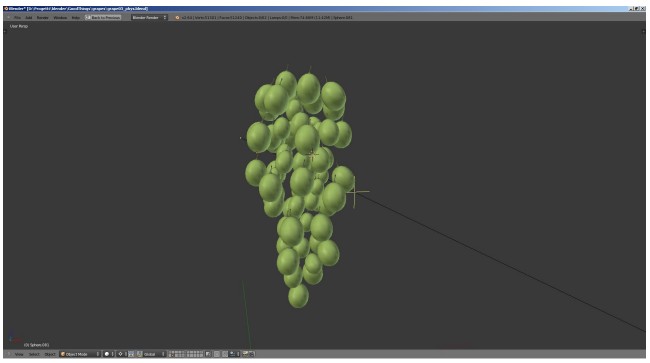


fig.06

pazienza e iniziare a disegnare il raspo in modo da collegare ogni acino precedentemente disposto ad un ramoscello. Potrebbe non essere necessario collegare tutti gli acini, visto che alcuni non saranno praticamente visibili. Questo dipende dal vostro modello. Ovviamente collegarli tutti non può certo far male.
Durante la creazione del raspo non siate avari

con il numero di vertici e variate spesso la scala degli stessi (ctrl+A) in modo da creare un bel look organico per i raspo. Ricordatevi che potete anche aiutarvi spostando gli acini ogni tanto. Inoltre, visto che abbiamo posizionato il pivot in cima al picciolo di ogni chicco, sara' molto semplice cambiare la rotazione e la scala in caso di compenetrazioni indesiderate.

Alla fine di questa operazione dobbiamo applicare i modificatori sulla mesh del raspo (vi consiglio di farvi una copia nel caso siano necessari degli aggiustamenti in seguito).
Aggiungete anche un modificatore displace con intensità' molto bassa e un altro subsurf. Come texture per il displace usate una normale cloud e aggiustate i settaggi finché il risultato non vi soddisfa.
A questo punto dovreste avere un bel grappolo (fig.07) pronto per ricevere materiali, luci e rendering!

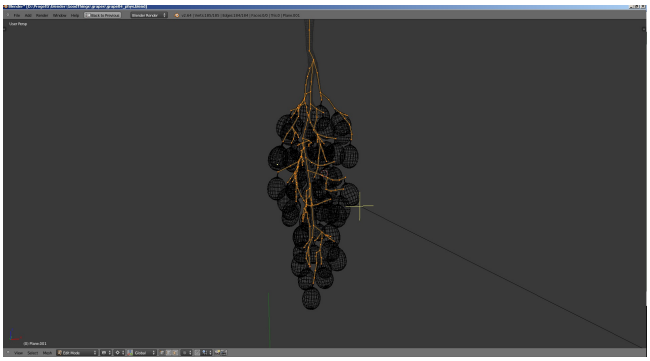


fig.07

SHADER ACINI E RASPO

Eccoci arrivati alla parte clou del tutorial. Il materiale dell'uva.
Partiamo proprio da questo. Visto che il materiale e' creato usando un buon numero di textures, qui di seguito vi riporto un'immagine con tutti i settaggi del materiale e delle textures usate. Di seguito commenterò i passaggi salienti.
L'uva che andremo a creare e' di quella verde. Se volete creare uva rossa potete ispirarvi a questo materiale, ma andranno cambiate alcune cose per quanto riguarda le textures.
Partiamo settando il colore diffuse come in figura. Usiamo anche un color ramp sempre sul

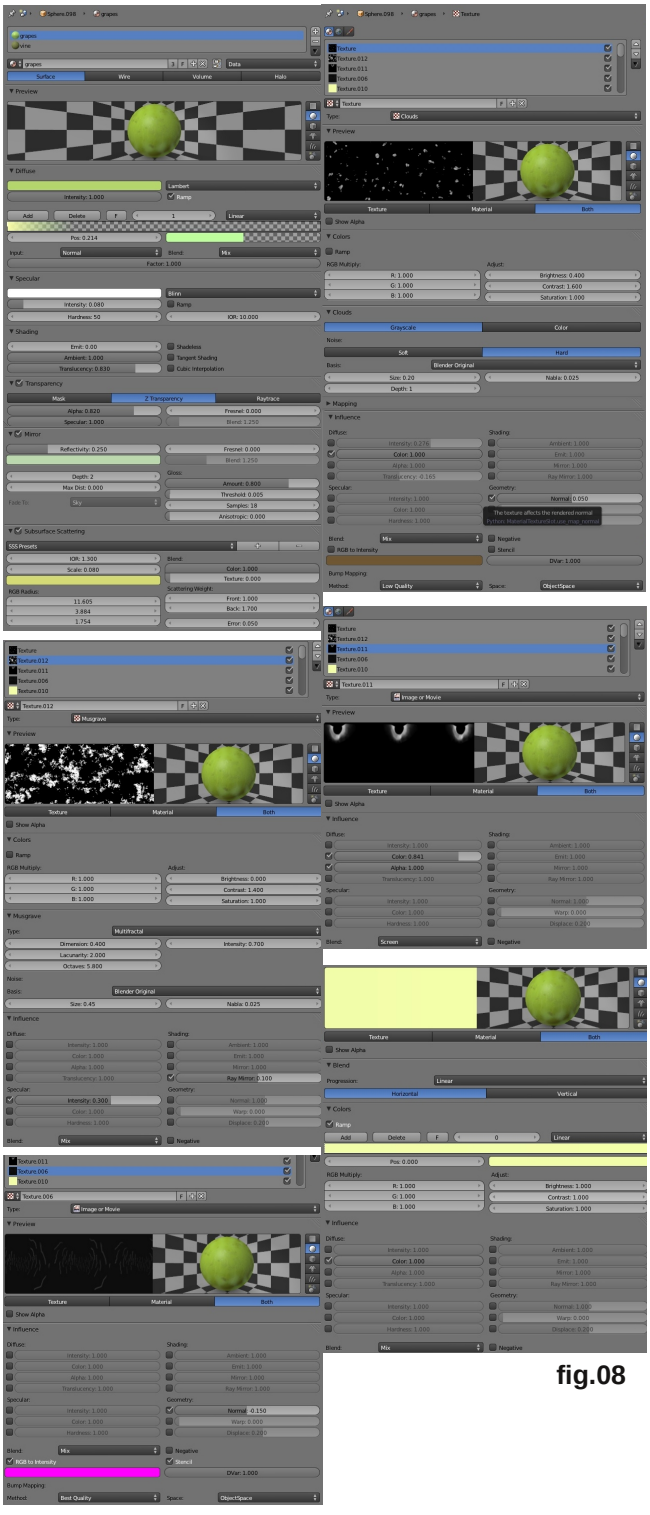


fig.08

diffuse con input normal. Come colore per la parte opaca useremo qualcosa di simile al diffuse ma più chiaro, mentre per l'alpha usiamo esattamente il colore del diffuse. Questo aiuterà l'sss creando un leggerissimo effetto di scattering della luce sui bordi degli acini.

Passate a settare specularità, translucenza, trasparenza e mirror come in figura.
Infine attivate l'sss. Ho usato il preset per la mela, cambiando il colore e aumentado il back scattering weight. All'inizio credevo che l'RGB radius avesse un rosso troppo intenso per l'uva. In seguito però ho visto che con questi settaggi si creava un piacevole effetto "giallognolo" e ho deciso di lasciare i valori di default. Per quanto riguarda il settaggio della scala, molto dipende da quanto grandi avete fatto gli acini. Il mio è alto circa una blender unit, picciolo escluso.

Occupiamoci ora delle textures. Le prime due sono procedurali e ci serviranno per creare rispettivamente le macchie marroni sugli acini e l'irregolarità della specularità tipica dell'uva (come se ci fosse un leggero strato di condensa sopra). Di nuovo i settaggi che vedete nell'immagine vanno bene per la scala dei miei acini. Se i vostri sono diversi dovrete aggiustare il tutto di conseguenza.
La terza texture è la prima immagine che abbiamo creato prima, il bianco nella parte alta dell'acino.
La quarta è la seconda immagine che abbiamo creato, le venature. Ricordatevi di mapparle entrambe sulle coordinate UV. Useremo la texture delle venature come stencil per un colore diverso. Aggiungete una quinta texture di tipo blend e usate una ramp con i colori e alpha uguali alle due estremità per ottenere un colore uniforme. Il colore dovrà essere simile a quello del diffuse, ma piu' chiaro e la texture va usata sul canale del diffuse.
Per regolare l'intensità di questa texture potete sia usare l'intensità su questo canale, sia giocare con brightness e contrast della stencil. Le parti bianche della nostra texture delle venature infatti saranno quelle che riveleranno il colore sottostante. Più' bianche sono e più' la texture sottostante sarà' visibile.

RASPO

Il materiale del raspo è relativamente semplice. Il colore è una via di mezzo tra un verde e un marrone.
Io ho usato un hex: 8C8F5B.
Azzerate la specularità (potete usare una piccolissima quantità di mirror abbassando il gloss, ma a fronte di un risultato di poco differente farete alzare i tempi di rendering sensibilmente). Attivate il receive transparent nelle sezione shadow (anche qui le differenze potrebbero non essere molte).

Come textures potete usare una musgrave abbassando il valore dimension e alzando octaves in modo da renderla più dura e irregolare. Usate un colore leggermente più verde di quello del diffuse e un po' di normal per dare variazione alla superficie.

Potete aggiungere altre textures di questo tipo per avere ancora più variazioni alla mesh. Non starò a dilungarmi troppo in quanto non ho curato particolarmente questo materiale.

ILLUMINAZIONE E RENDERING

Il setup luci è molto semplice. Abbiamo una spot come fonte principale. Ho usato delle buffer shadows molto morbide (15) e un leggero halo (non esagerate con questo effetto). Fatela abbastanza potente (io ho usato 5) e non siate avari con la distanza. Il colore è leggermente tendente al caldo.
Con direzione praticamente opposta alla spot ho usato un hemi molto debole (0.1) e con un colore leggermente più freddo.
La disposizione la potete fare a vostro gradimento. Di massima fate puntare la spot dall'alto in basso e posizionate l'hemi di conseguenza. Più la spot si trova dietro al grappolo rispetto alla camera, più l'sss degli acini

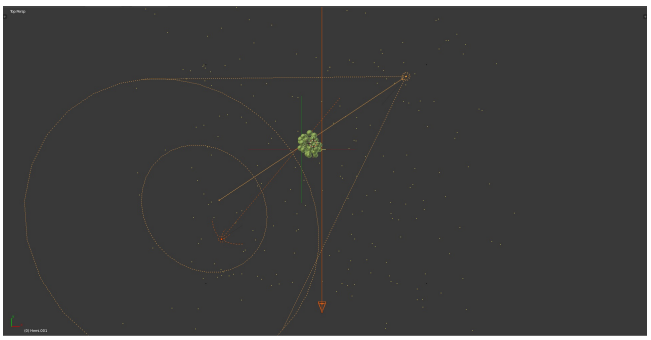


fig.09

verrà esaltato (fig.09).
I settaggi world sono praticamente di default.
Non ho neache cambiato il colore dello sfondo, ma ho semplicemente disattivato il rendering dello sky nelle opzioni di rendering. Niente AO o Enviroiment.
Infine ho usato un sistema particellare con materiale halo (Alpha: 0.05, Size 0.05, hardness 0 e un colore giallino) per creare un po' di atmosfera e ho fatto un leggero compositing sull'immagine (fig.10).

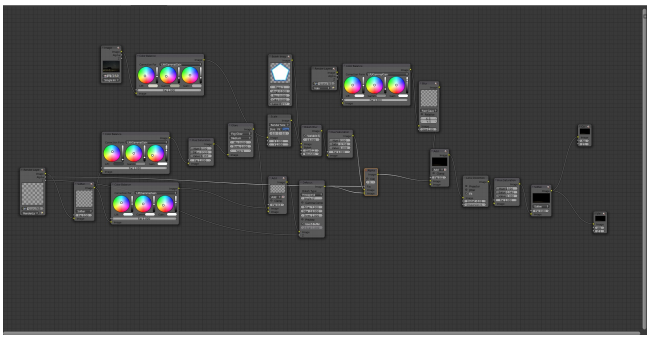


fig.10

E siamo arrivati alla fine di questo tutorial. Spero di non avervi annoiato e che abbiate imparato qualcosa... Vi ringrazio per l'attenzione!

Ci vediamo su Blender Italia!

Fotoinserimenti semplici

Impariamo ad usare BLAM per gli inserimenti fotografici

di Alberto Lipari (orom)

Introduzione

Il Blender Camera Calibration Toolkit (di seguito BLAM) è presente già da qualche release, fra gli addons indispensabili da installare per chi intende utilizzare il nostro software open per la visualizzazione in ambito design e architettura. Questo tool permette infatti di calibrare molto semplicemente la telecamera virtuale di Blender secondo le coordinate prospettiche di una immagine da noi caricata come sfondo. Si presta quindi perfettamente per la creazione di foto-inserimenti con i nostri modelli 3D.
Il BLAM è scaricabile da
https://github.com/stuffmatic/blam/downloads

Nota sulla compatibilità di versione

L'ultima versione stabile di Blender 2.65a risulta perfettamente compatibile con la versione stabile di BLAM 0.4 . Stessa cosa non si poteva dire per le prime experimental builds della prossima 2.66 che a causa di alcune modifiche alle api python non risultavano più compatibili con alcuni addon (tra cui il BLAM purtroppo). Il bug è stato però

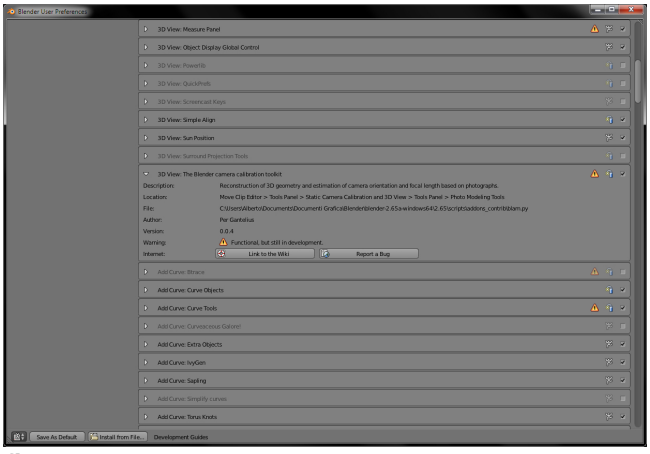


fig.01

segnalato e prontamente risolto da Campbell Barton.
La nuova versione è scaricabile da
https://github.com/stuffmatic/blam/raw/master/src/blam.py

Acquisizione del materiale fotografico

Per eseguire un buon foto-inserimento è fondamentale disporre di un buon materiale fotografico riguardante l'ambientazione. Tra

questo infatti dobbiamo scegliere la o le immagini in cui inserire il nostro modello. Vi ricordo, a tal proposito, che per il corretto funzionamento del BLAM è necessaria la presenza sia nella foto che nella scena 3D di oggetti “reference” con spigoli fra loro ortogonali (murature, cassonetti, segnaletica, mobilio ecc.). Da essi bisognerà infatti desumere le coordinate prospettiche fondamentali per ricreare l'allineamento corretto della telecamera.
Di grande utilità è anche prendere nota delle dimensioni lineari di questi oggetti (altezza, larghezza, profondità) per rispettare nel modello le proporzioni esatte.
Bisogna ricordare inoltre di curare l'illuminazione e la composizione delle foto applicando le eventuali correzioni prima di procedere alla loro importazione in Blender.

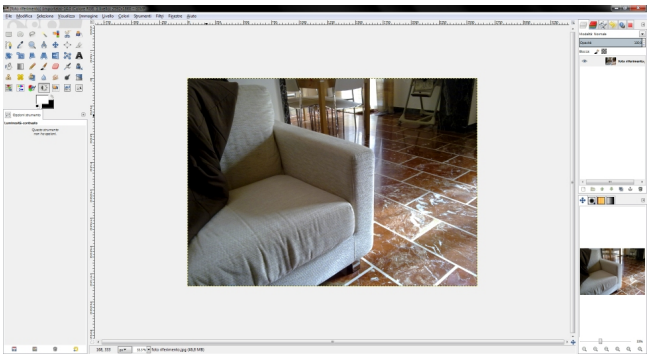


fig.02

Modellare l'ambientazione

Scelta la foto si passa alla modellazione della scena. Innanzi tutto bisogna riprodurre il più fedelmente possibile (almeno per quanto riguarda forme e dimensioni) l'ambientazione in cui verrà inserito il modello. Nell'esempio proposto ho cercato di riprodurre la stanza e il divano che fanno da sfondo e supporto al modello 3D del tavolo/vassoio che vogliamo foto-inserire.

Chiaramente bisogna prestare attenzione a che anche l'illuminazione naturale e/o artificiale sia il più possibile accurata e somigliante all'originale. Per questo scopo l'ideale sarebbe poter usufruire di una HDRI catturata nella stessa sessione di acquisizione delle foto di riferimento.

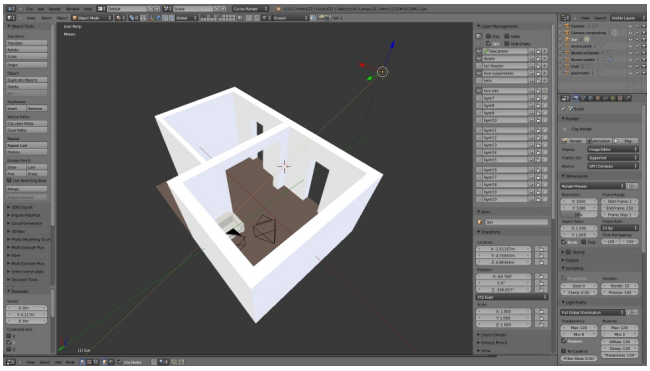


fig.03

Tuttavia anche con la Sun Lamp, opportunamente regolata, e la Sky Texture di Cycles si possono ottenere dei buoni risultati.

Modellare il soggetto

Completata la modellazione dell'ambiente si passa a quella dell'oggetto vero e proprio. Quando si opera in campo architettonico normalmente si dispone già di reference 2D accurate, quindi è piuttosto semplice rispettare le proporzioni e la posizione nell'ambiente.
Nell'esempio, pur non avendo reference, ho cercato di rispettare delle misure realistiche nelle varie parti che compongono il modello, tramite le unità metriche impostate in Properties / Scene / Units.
Al termine ho anche aggiunto degli oggetti extra per rendere più naturale la scena ed ho impostato e assegnato i materiali.

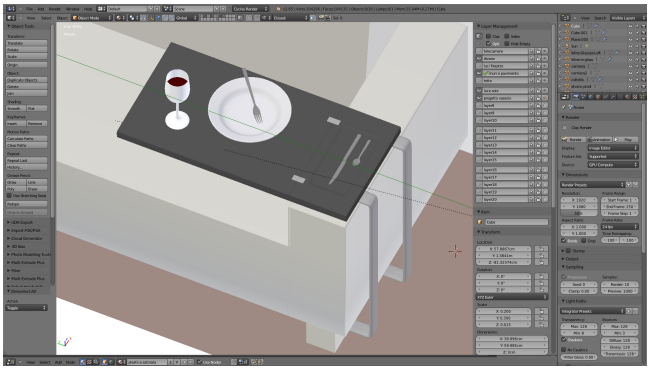


fig.04

Creare la telecamera e i riferimenti con Grease Pencil

Creare una telecamera dedicata al compositing e posizionarla (Shift+F) in modo da inquadrare la

scena in maniera simile a quella della foto di riferimento.

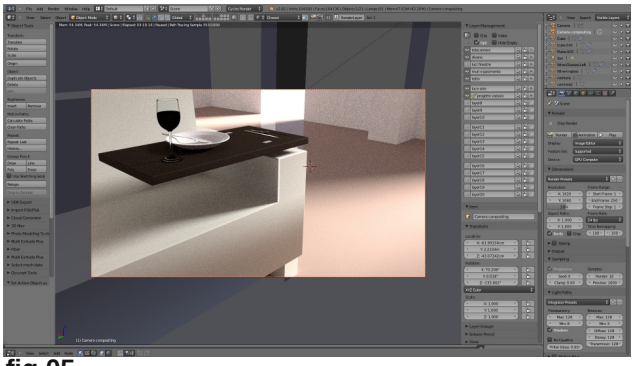


fig.05

Adesso possiamo iniziare il lavoro di compositing vero e proprio.
Apriamo all'interno della finestra Movie Clip Editor la foto di riferimento. Nel pannello di destra attiviamo il Grease Pencil e creiamo 3 nuovi livelli in corrispondenza degli assi X ,Y e Z della scena. Per tale scopo può essere utile associare il colore di ciascun layer con quello dell'asse di riferimento nello spazio viewport 3D. Tracciamo quindi delle linee rette aiutandoci con gli oggetti di riferimento presenti nella foto: piastrelle, mobilio, spigoli delle murature ecc. (Ctrl+D + LMB).

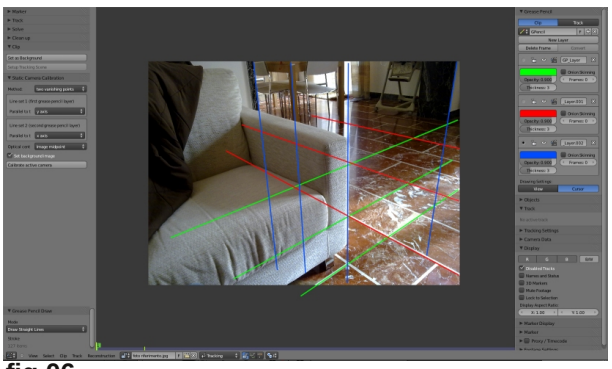


fig.06

I parametri del Camera Calibration

Nel pannello a sinistra troviamo la scheda Static Camera Calibration in cui dobbiamo impostare la corrispondenza fra gli assi disegnati col Grease Pencil e quelli della scena. Inoltre bisogna impostare il metodo di individuazione del centro ottico dell'inquadratura. Nell' esempio avendo rilevato anche gli assi sulla Z (rette azzurre) ho scelto l'opzione From 3rd vanishing point.

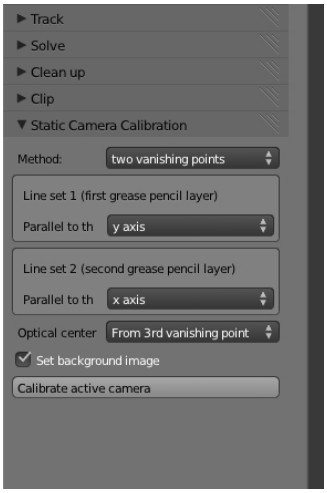


fig.07

Spuntando l'opzione Set background image possiamo far si che la foto di riferimento sia impostata automaticamente come sfondo della vista telecamera. Ciò sarà utile per la verifica e l'eventuale aggiustamento manuale dell'inquadratura.
Per settare la telecamera basta premere sul pulsante Calibrate active camera.

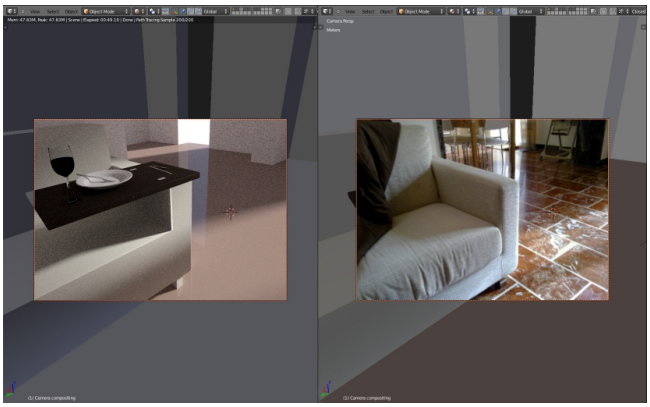


fig.08

Rendering

A questo punto abbiamo la nostra scena inquadrata con la corretta calibrazione. Noteremo anche che la dimensione della telecamera (e quindi del rendering) in Properties / Render / Dimensions è divenuta uguale a quella della foto di riferimento utilizzata. Qualora fossero necessari degli aggiustamenti di posizione si può intervenire dalla stessa vista telecamera effettuando dei piccoli spostamenti in parallello con il MMB dopo aver attivato la Fly Navigation (Shift+F). Attenzione a non “girare” invece la telecamera su se stessa: si potrebbe modificare negativamente l'inquadratura impostata!
Controllando attraverso l'uso del comando Background Image la coincidenza fra foto di riferimento e scena 3D possiamo passare alla fase del rendering vero e proprio. Prima però dobbiamo agire sulla “trasparenza” al rendering

degli oggetti di riferimento che abbiamo utilizzato per la calibrazione della telecamera, togliendo la spunta da Camera in Properties/Object/Ray Visibility.

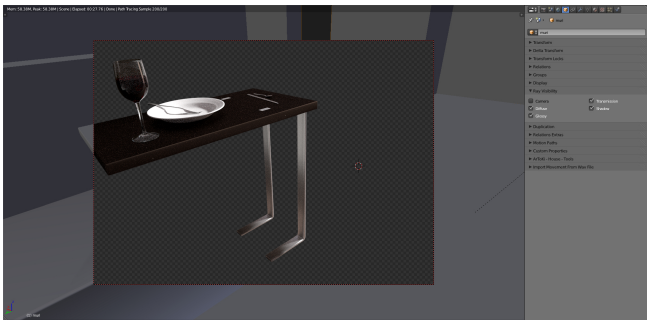


fig.09

Rendiamo trasparente anche l'environment del render spuntando la casella Transparent in Properties/Render/Film e quindi procediamo al rendering.

Compositing in GIMP

Ottenuta l'immagine del modello e salvata in formato PNG – RGBA si può procedere al compositing con la foto di riferimento. Esistono molteplici modi, tutti ugualmente validi per unire render e foto, anche attraverso l'utilizzo avanzato dei nodi nel compositor di Blender. Nell'esempio ho invece utilizzato il celebre programma di fotoritocco GIMP per eliminare innanzi tutto le parti del modello nascoste dagli oggetti reali (il divano). Quindi utilizzando diversi layer ho dipinto le ombre proiettate dal tavolino sul sedile del divano. Ho eliminato anche il residuo di rumore nel render utilizzando il filtro Smacchiatura. Per finire ho regolato l'equilibrio cromatico e l'illuminazione/contrasto del render per migliorarne la somiglianza con l'illuminazione della foto di riferimento.

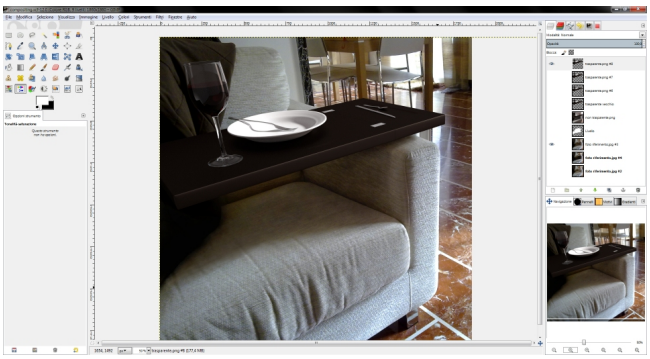


fig.10

Diseguito potete osservare il risultato finale con l'immagine opportunamente ritagliata.

fig.11

Conclusioni

Il BLAM non esaurisce comunque qui le sue potenzialità in quanto permette di realizzare ulteriori operazioni. Chi fosse interessato ad approfondire la questione può visitare il seguente indirizzo
https://github.com/stuffmatic/blam/wiki/User%27s-guide
Spero che questo tutorial vi sia utile.
Un abbraccio e buon lavoro!

Alberto Lipari
orom@virgilio.it
www.alcreations.altervista.org

Casa del futuro

Contest per scegliere la prossima copertina di B.M.I.

Il tema del contest per la copertina del prossimo numero di B.M.I. è "Casa del futuro".

Immaginatevi dove vivremo tra dieci, cento o mille anni e disegnatela per creare l'immagine di copertina del prossimo numero di B.M.I.

Il regolamento è sempre lo stesso:

- Le immagini devono essere eseguite principalmente con Blender.
- Si possono eseguire anche più lavori, ma ricordate che conta la qualità non la quantità.
- Sono permessi motori di rendering esterni.
- Sono permessi altri programmi esterni ma solo per ritocchi di postproduzione ed a patto che siano Open.
- Le immagini devono avere queste dimensioni: 1024*1448 px o superiore, basta rispettare la proporzione A4.
- Le immagini possono essere postate come Wip su Blender.it o Kino3d.com con la dicitura [BMI17] prima del titolo.
- I lavori finiti devono essere inviati ad alfonso.annarumma@gmail.com **entro il 9 giugno 2013**.
- Ricordate di specificare il titolo dell'immagine e il nome che volete usare come firma (nick e/o nome vero).

- Il vincitore avrà l'onore di vedere pubblicata la sua immagine sulla copertina di Blender Magazine Italia.
- Il vincitore potrà scrivere un piccolo Making Of del suo lavoro che sarà pubblicato sulla stessa rivista elettronica. L'articolo non è obbligatorio.

Al termine della data di consegna, verrà aperto un sondaggio affinchè, chi vuole, possa votare, tra le immagini partecipanti, quella che vorrebbe vedere sulla copertina del prossimo numero di Blender Magazine Italia.

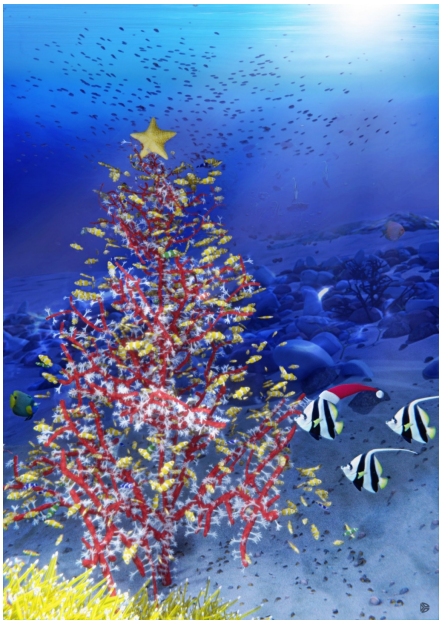
Gallery dell'ultimo Contest

L'ultimo contest per scegliere la copertina di B.M.I. è stato realizzato con una fusione con il contest natalizio di Blender.it
Il titolo del concorso era "Natale a..." e numerosi sono state le immagini in gara. Tramite le votazioni sul forum di Blender.it e quelle di una giuria "premium" sono stati scelte le immagini migliori.

Primo premio della giuria
a Seregost
(lettore mp3 Apple Ipod Shuffle)

Primo premio del forum
ad Ego
(televisore Led 19", zaino da montagna, telefono skype USB)

Secondo premio del forum
a Riky
(lettore mp3 Apple Ipod Shuffle, telefono USB skype)



Terzo premio del forum
a Dacam3d
(lettore mp3 2gb, borsa Sport Diadora)

SANDY ISLAND

AUTORE: Luigi Maggio
EMAIL: luis_may86@libero.it
WEBSITE: luigimaggio.altervista.org

FOGLIA

AUTORE: Marco Crippa (krypt)
EMAIL: thekrypt77@gmail.com
WEBSITE: krypt77.altervista.org

PRESEPE

AUTORE: Luigi Russo
EMAIL: luigirusso_87@hotmail.it

BAGNO

AUTORE: perepepe77
EMAIL: saccoman.enrico@libero.it

DUCATI SOGNO

AUTORE: Mauro Bonecchi - meda36

EMAIL: bullx@meda36.it

Blender Magazine Italia
numero 16
anno 2013

Responsabili:
Alfonso Annarumma (Anfeo)
Luca Pinciani (Sinistar)

Collaboratori:
Andrea Fiocca (gikkio)

Grafica:
Davide_G

Siti di riferimento:
www.blender.it
www.kino3d.com
www.blender.org

Software utilizzati:
Blender
Scribus
Pdftk
The Gimp
LibreOffice

Vuoi collaborare con noi?

Scrivi un articolo per Blender Magazine Italia

Ti stiamo aspettando!

Le indicazioni per scrivere un buon articolo sono semplici:

- Scrivere un documento di testo apribile da LibreOffice o OpenOffice senza immagini ma con i riferimenti per inserire l'immagine corretta nella posizione esatta.

- Creare una cartella con le immagini e nominarle come scritto nel testo dell'articolo (ad es. fig1.jpg, fig2.png).

- Creare una immagine per l'header dell'articolo.

- Fornire (in modo facoltativo) descrizione personale, contatti e sito internet dell'autore.

Invia l'articolo all'indirizzo
alfonso.annarumma@gmail.com

SITO WEB

Tutti i numeri di
Blender Magazine Italia
sono disponibili
per il download gratuito
all'indirizzo
www.BlenderMagazineItalia.it