

# Appunti Liberi

**Manuel Bastioni, mbastioni@tiscalinet.it**

Versione 0.01-alpha / 2003-01-03 22:39:12

---

*Questo documento e' rilasciato secondo la licenza FDL per maggiori informazioni <http://www.gnu.org/copyleft/fdl.html>*

*Copyright (c) 2002 Manuel Bastioni.*

*Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation;*

---

## 1. Introduzione

- 1 Licenza Blender Appunti Liberi
- 2 Cenni storici
- 3 Features
- 4 Requisiti di sistema
- 5 Installazione
- 6 Dove trovo l'ultima versione di questi appunti?

## 2. Interfaccia

- 1 Alcune convenzioni
- 2 La schermata iniziale
- 3 Gli Screens e le Windows
- 4 Elementi di partenza
- 5 La modalita' Trackball: attorno all'oggetto.
- 6 Le 3 viste classiche: sopra, davanti e di lato
- 7 View modes
- 8 View Object
- 9 Layers
- 10 Selezioni
- 11 Oggetti attivi e oggetti selezionati.

## 3. Editing di base

- 1 Le primitive
- 2 Mesh
- 3 Le Metaballs
- 4 Le NURBS
- 5 Trasformazioni di base: muovere, ruotare,
- 6 Translazione
- 7 Rotazione

- 8 Dimensioni (Scaling)
- 9 Trasformazioni attivate dai movimenti del mouse
- 10 Snap
- 11 Lo snap a griglia
- 12 Settaggio della griglia e snap cursore

## 4. Material editing

- 1 UV mapping
- 2 Caricamento della bitmap
- 3 Tipi di proiezione UV
- 4 UV editing
- 5 Rendering di mappe UV
- 6 Effetti speciali
- 7 Sfumatura wire-solid

## 5. Editing avanzato

- 1 Oggetti ed EditMode
- 2 Shrink/Fatten (ALT-S)
- 3 Il pulsante Hash
- 4 Il pulsante Xsort
- 5 Creazione di facce
- 6 Operazioni booleane
- 7 Lavorare con le NURBS

## 6. Animazione

- 1 Animazione di base: keyframes
- 2 Navigazione tra i frames
- 3 Utilizzo della cinematica inversa
- 4 Lo scheletro di IKA
- 5 La IPO windows
- 6 Animazione ciclica
- 7 RVK (Relative vertex key)
- 8 Uso delle RVK
- 9 Sincronizzazione e morphing delle RVK

## 7. Radiosity

- 1 Cosa é la radiosity
- 2 Qualche dettaglio tecnico
- 3 I tre sistemi di rendering
- 4 Il rendering scanline
- 5 Lo scanline Phong e Goraud
- 6 La discretizzazione della Radiosity
- 7 L'equazione della radiosity

- 8 Materiali emittenti
- 9 Uso dei pulsanti
- 10 Uso delle texture
- 11 Vantaggi e svantaggi
- 12 Webografia

## 8. Python script

- 1 Introduzione
- 2 L'interprete Blender
- 3 La text-window
- 4 Operazioni in text-window
- 5 L'importanza dei rientri
- 6 Le basi
- 7 I Moduli
- 8 I moduli Builtin
- 9 Il modulo sys
- 10 Funzioni matematiche
- 11 Funzioni generali
- 12 Strutture di dati
- 13 Accesso alle liste
- 14 I cicli in python
- 15 Gli oggetti di Blender
- 16 Le funzioni standard di Object
- 17 il modulo NMesh

## 9. Game Engine

- 1 Introduzione
- 2 Le opzioni del Game engine
- 3 Le opzioni della finestra info
- 4 Bottoni nella finestra real-time
- 5 Bottoni nella finestra dei materiali
- 6 Proprieta' dell'oggetto
- 7 Introduzione ai LogickBricks e Sensori
- 8 Controllori & Esecutori
- 9 Python scripting per il GameEngine

## 10. Plugins

---

### 1. Introduzione

Manuel Bastioni, mbastioni@tiscalinet.it  
2002-12-11 22:52:32

Contenuto della sezione

# 1.1 Licenza Blender Appunti Liberi

## Blender Appunti Liberi

GNU Free Documentation License

Version 1.1, March 2000

Copyright (C) 2000 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

### 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

## 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3. You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## 3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition.

Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

#### 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.

I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section.

You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice.

These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this

License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

#### 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

#### 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate.

Otherwise they must appear on covers around the whole aggregate.

#### 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4.

Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the

original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

## 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

## 1.2 Cenni storici

Blender Creator e' un programma nato nel gennaio del 1996 per la creazione di modelli e immagini per giochi su console NeoGeo. Nel gennaio del 1998 venne pubblicata la versione 1.23 SGI, ed IrisGL, mentre la versione 1.30 implemento' anche la piattaforma FreeBSD, con il porting ad OpenGL ed X. La Musa, nel numero 15 della versione italiana, pubblicava gia' che

*Blender 1.37*

*Proprio perché si tratta di un software realizzato all'interno di uno studio che fa animazione di alta qualità, Blender ha dimostrato di essere uno strumento di progettazione molto versatile e veloce. Il software ha un tocco di personalità che ne rende unico l'approccio al mondo delle tre dimensioni. Blender può essere utilizzato per creare spot per TV commerciali, visualizzazioni tecniche, grafici contabili e finanziari, per fare un po' di morfing o per progettare interfacce d'utente. Con questo software si possono costruire e gestire ambienti complessi in maniera molto facile. Il rappresentatore è versatile ed estremamente veloce. Tutti gli elementi su cui poggiano le animazioni (curve e chiavi) sono implementati in maniera perfetta.*

*La versione 1.37 aggiunge le applicazioni UV per NURBS e corregge una serie di precedenti errori.*

Fallita la ditta che lo produceva, uno dei programmatori, Ton Roosendaal ha continuato a sviluppare il software in modo autonomo sotto la bandiera della Not A Number, che venne fondata nel Giugno del 1998. Nel settembre dello stesso anno, Blender 1.4 venne portato anche su Linux e Sun. Nel novembre 98 insieme alla 1.5 uscì anche la prima edizione del manuale. Nell'aprile del 1999 venne istituita la C-key, per cui le nuove caratteristiche venivano bloccate a meno che non venisse comperata la licenza, al prezzo di 95 dollari.

Con la 1.6 finalmente fu rilasciata la prima versione per PC, insieme a quella per BeOS. Con la 1.8 Blender torno' ad essere freeware, poiche' la C-key fu abolita. Nell'agosto del 2000 uscì il game

engine, che permette di creare giochi interattivi in 3D. Nel dicembre dello stesso anno venne implementato un nuovo engine ed una nuova API per il Python, il potente linguaggio di script che permette un'espansione senza limiti delle sue capacita'. La 2.20 ha introdotto il potente Character animation system, in grado di muovere personaggi in modo realistico. Infine con il Web plugin e' possibile usare i file di Blender in maniera interattiva senza bisogno di Blender stesso, ma solo tramite i browser web, on o online.

### **Blender OpenSource**

Nel marzo del 2002 la NaN dichiara fallimento. Si tratta sicuramente del piu' grave momento nella storia di Blender. Con tono dismesso, nell'unica pagina rimasta attiva, la NaN comunicava:

#### ***Statement of NaN technologies as released on march 14th 2002:***

*NaN Technologies closes down*

*The shareholders and directors of NaN Holding BV, owners of Blender, have decided to terminate all activities of NaN Technologies BV and apply for its bankruptcy at the Amsterdam court. It means that effective today, all technology development and website activities around Blender will be frozen.*

*This page is the only page available on the website, until further notice. We sincerely apologize for the inconveniences to all website visitors and Blender community members. The digital media market has shown that it is not quite ready for the Blender technology offering, both for the Internet and for wireless applications. Continuation of these products, including the Creator, Publisher and our wireless initiative, will require additional investment of human and financial resources.*

Si e' trattato di un evento doloroso, e' vero, ma ha dato una svolta alla storia della Computer Grafica Open Source. Infatti l'enorme comunita' di utenti, dopo un primo momento di dispersione, si e' riunita e riorganizzata all'interno di un nuovo sito, [www.elysiun.com](http://www.elysiun.com).

Qui Ton, senza perdersi di animo, il 18 aprile del 2002 scriveva sul forum:

- *get a ticket to Siggraph, doing a campaign there (donate!)*
- *make t-shirt (FREE BLENDER FUND), for people to wear at Siggraph (tell me, who goes?)*
- *press, slashdot, other news sites. (everyone: go ahead!)*
- *find a company willing to allow me space at their Siggraph booth (who knows some?)*
- *find internet millionaires or companies willing to become sponsor, define interesting benefits for them. (contact me!)*
- *getting back old blender.nl webservices*

Questa reazione, nel giro di poco tempo, prese la forma di una scelta storica: Blender sarebbe dovuto diventare OpenSource, e rilasciato sotto licenza GNU GPL!

Non si trattava pero' di un'impresa facile: occorreva pagare ben 100.000 euro. Il 5 Luglio Ton scriveva:

*Today the shareholders of NaN Holding have reached an agreement on the outlines for a new future for Blender. In general it means that a non-profit organisation (the Blender Foundation) will be enabled to execute its plans, including Blender development as an 'open source' or 'free software' project.*

*Details of this agreement will be studied on and negotiated during the next week.*

*What the NaN shareholders and the Foundation agree on:*

- *publishing the full Blender sources, including old and new development, under a GNU GPL license.*
- *the Foundation will pay an initial fee of 100k euro for this (98k USD)*
- *the Foundation can exploit the Blender website and re-establish e-shop services*
- *NaN Holding will be sufficiently enabled to (re)start business in the future, for example licensing derived technology or professional services.*

*NaN Holding recognizes that, giving all circumstances and the current economic situation, moving on with Blender to this next stage will be the most beneficial thing to do, to protect past investments, but also to respect everything that has been realized until now by the NaN companies and the world-wide user community.*

*I am very happy we were able to make this tough decision, hopefully it will become a historical step.*

*Details on the activities to gather funding will be made public here soon. Stay tuned!*

A questo punto avvenne una specie di miracolo. L'enorme comunita' di Blender, innescata dalla possibilita' di far resuscitare Blender, riesce a raccogliere l'enorma somma esclusivamente con le donazioni. Ma quello che e' piu' impressionante e' la rapidita' dell'evento. Appena il 7 settembre 2002, alle ore 13.00 CET, Ton annuncia:

*We've met our target, collecting sufficient money to pay for releasing the Blender sources! Thanks everyone!*

## 1.3 Features

### **Caratteristiche generali**

Modellazione di poligoni meshes, curve, nurbs, testo, metaballs  
Animazione con keyframes, motion curves, morphing, cinematica inversa  
Deformation lattices e skeletons  
Sistema particellare  
Rendering: solid, transparent, halo/lensflare.  
Sequence editing di immagini e effetti di post produzione  
3D view con rotoscopia animata  
Object oriented data system

### **Layout**

Layout completamente personalizzabile  
3D window wireframe solid OpenGL lighted-rendered.  
Scelta della quantita' di particolari visualizzati durante la lavorazione.  
Bottoni ridimensionabili  
Finestra di animazione per curve-chiavi window  
Finestra schema a diagrammi  
Finestra per editing di sequenze video  
Completa gestione dei file interna al programma

### **Files**

Salvataggio del lavoro in un unico file  
Possibilita di usare altri blender file come libreria di materiali-oggetti  
Scrittura e lettura di file Targa, JPEG, Iris, SGI movie, Amiga IFF  
Import-export di file Open Inventor, dxf, Videoscape e VRML 1.0

### **Luci**

Local lights, spotlights, hemispheres, sun.  
textured lights, spotlights  
Shadow buffer system  
Selective lighting per oggetti singoli

### **Render**

Unified renderer  
Rendering con salvataggio diretto  
Possibilita di effettuare il rendering da qualsiasi modalita'  
3 rendering layers: per solid, transparent e halo's.  
Jittering per un perfetto antialiasing  
Risoluzioni sino a 4096x4096  
Field rendering and pre-gamma correction per il miglior output video  
Radiosity  
Panorama rendering  
Plugins per textures e post production

### **Animazione**

Mesh Morphing

Motion paths: Bezier curves, B-splines or polygons. - Motion curves:

XYZ translation-rotation-scaling

Motion keys: transformations fixed at a frame - vertex key framing for morphing

Inverse kinematics

Animation curves for light, materials, textures, etc.

Inserimento automatizzato delle keyframes per le animazioni e per gli oggetti.

Character Animation e Skin deformation

### **Meshes**

Utilizzo dei poligoni quadrati e triangolari

Extrude, spin, screw, bend, subdivide, etc.

Colorazione vertici in realtime (3d vertex paint)

Catmull-Clark suddivisione delle superfici.

Proportional Editing tool

### **Curve**

Bezier, Bspline, polygons

automatic 'hole' detection.

qualsiasi curva puo essere utilizzata per effettuare un bevel(smusso)

### **Mappatura**

Colorazione delle mappe in 3D realtime (3d Paint)

Environment mapping

UV editor and FaceSelect

### **Game Engine**

Attori, attuatori, sensori di prossimita'

simulazione dinamica delle leggi fisiche, python script

## **1.4 Requisiti di sistema**

La visualizzazione ombreggiata in real time viene ottenuta grazie alle librerie di sistema OpenGL (Open Graphic Library) per cui e' necessario che sul sistema si trovino i file Opengl32.dll e glut32.dll, che si trovano anche sul CD di installazione di Windows 95/98/2000.

Blender e' in grado di funzionare anche su di un antico 486DX50 con almeno 8 mega di ram ed una scheda grafica primitiva tipo Cirrus 5422.

Tuttavia per lavorare decentemente sono necessarie al minimo 32 mega di ram.

## **1.5 Installazione**

l'installazione sotto Windows e' semplicissima. Una volta scaricato il file zippato

blenderX.XX-windows.zip e' sufficiente decomprimerlo dentro una qualunque cartella, ad esempio

C:/Programmi/Blender. Bastera' poi cliccare

due volte sul file blender.exe per far partire il programma. E' consigliabile creare un collegamento sul desktop. Blender non utilizza file tipo .ini, ma all'apertura carica un file di default chiamato .B.blend.

Per la configurazione del python vedi

Introduzione al python|C8Q2

## 1.6 Dove trovo l'ultima versione di questi appunti?

Questi appunti nascono da un lavoro iniziale di Manuel Bastioni, che in origine aveva solo intenzione di riordinare le proprie idee riguardo l'uso di Blender.

Sarebbero rimasti inutilizzati se non ci fosse stato l'intervento di Kino, che con grande entusiasmo decise di trasformarli nello strumento interattivo di

<http://www.kino3d.com/Blender-AL>

Kino3d.com Appunti Liberi

Su kino3d risiede dunque il laboratorio on line in cui i vari autori lavorano in tempo reale per aggiornare ed ampliare la documentazione. Chiuque e' invitato ad una seria partecipazione. E' ovvio che la versione degli Appunti e' in continua evoluzione, e propbabilmente, gia' mentre state leggendo, qualcuno ha gia' aggiunto nuove informazioni ...

---

## 2. Interfaccia

2002-12-12 01:36:06

Contenuto della sezione

### 2.1 Alcune convenzioni

Sebbene l'interfaccia di Blender possa all'inizio apparire complessa e poco familiare, ci accorgeremo ben presto che la disposizione degli strumenti e' perfettamente logica, e straordinariamente personalizzabile; ad esempio, avviando blender tramite linea di comando, con le opzioni tipo:

```
blender -p 0 0 1024 768
```

si possono impartire le dimensioni iniziali della finestra, che in questo esempio vengono settate a 1024 per 768.

La verita' e' che l'interfaccia di Blender non e' rivolta ad un utente dilettante ed e' studiata per ovviare alla situazione tipica una miriade di finestre e finestrelle indipendenti e fastidiose che occupano in modo caotico lo spazio consentito: essa e' ottimizzata per il massimo rendimento e per avere in ogni momento tutto sotto controllo, anche se questo va a discapito della intuitivita'. Su molti articoli presenti in rete ho trovato questa frase, che racchiude un po' la filosofia dell'utilizzo di Blender: "metti una mano sul mouse ed una sulla tastiera".

Infatti molte funzioni sono accessibili direttamente dalla tastiera, che svolge un ruolo primario nell'utilizzo del programma. Blender e' stato pensato per utilizzare entrambe le mani e il tasto centrale del mouse. Diciamo fin da ora che sotto windows il tasto centrale, qualora non sia configurato, puo' essere emulato premendo il tasto destro del mouse + il tasto ALT

#### **Importante:**

D'ora in poi, per i tasti del mouse useremo espressioni come *Rmouse*, *Lmouse* e *Mmouse*, (dalla nomenclatura inglese Right-mouse Left-mouse e Middle-mouse). Per i pulsanti del tastierino numerico useremo invece la nomenclatura *NUMPAD1*, *NUMPAD2* etc...Sempre con riferimento al tastierino numerico indicheremo anche lo zero con *ZEROKEY*, il + con *PADPLUS* ed il - con *PADMINUS*. Per i tasti freccia invece parleremo di *LEFTARROW*, *RIGHTARROW*, *UPARROW*, *DOWNARROW*. Il tasto *shift*, usato per fare le maiuscole, rappresentato sulla tastiera con una spessa freccia rivolta verso l'alto, sara' chiamato *SHIFT*.

*ALT* e *CTRL* si scrivono uguali ai loro tasti.

I tasti semplici, coma A,B,C,D etc... saranno chiamati *Akey*, *Bkey*, *Ckey*, *Dkey*, etc...

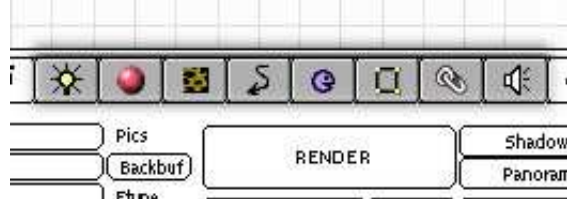
La barra spazio verra' chiamata *SpaceBar*...

## 2.2 La schermata iniziale

All'apertura di Blender in Windows partira' prima un finestrella DOS, detta *console*, particolarmente utile per il debug degli script python; dopo pochi istanti si aprira' Blender vero e proprio, con una window a tutto schermo divisa in tre parti (3 frames): Il sistema e' simile a quello dei frames dei file html.

Sotto Linux invece, cliccando direttamente sull'icona, Blender parte senza console. Per avere la console occorre avviarlo da linea di comando.

Ogni window ha un *header*, cioe' una barra di intestazione con diverse funzionalita'.



Esempio di header: quella della 3D window

Ecco un primo assaggio della personalizzabilita' di Blender: potete scegliere se posizionare tale header all'estremita' superiore o inferiore della propria window: cliccate sopra alla barra di intestazione con Rmouse e poi sul pulsante *Switch Header* che comparira' nella finestra pop-up.

Blender 2.x offre una nuova possibilita': quella di eliminare completamente l'header. Per ripristinare la visibilita' di un header basta scegliere l'opzione adeguata che si presentera' cliccando sul suo bordo

(normalmente, come vedremo tra breve, cliccando sul bordo si scelgono le opzioni per creare o eliminare nuove finestre, ma a queste si aggiunge automaticamente quella del ripristino dell'header, qualora ci si trovasse su una finestra priva di intestazione) con il tasto destro.



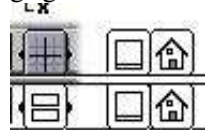
In alto trovate la prima delle tre window: si tratta della *Info Window*, come indica la l'icona 'i' a sinistra nella sua header. Questa finestra contiene, appunto, le informazioni riguardanti il progetto, come lo screen su cui

stiamo lavorando, il numero di vertici e di facce dell'oggetto che stiamo editando ecc. . . Dalla versione 2.x in poi la Info contiene anche gli usuali menu' a tendina a cui siamo abituati in ambienti Dos e Windows.

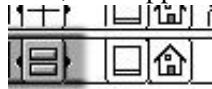


La Info Window ha una parte nascosta, per non occupare spazio prezioso del vostro schermo, ma la potete visualizzare trascinando la linea spessa che delimita il suo bordo inferiore.

Al centro trovate la 3D Window, come indica l'icona (sempre sulla sinistra della sua header), di una griglia divisa in quattro parti.



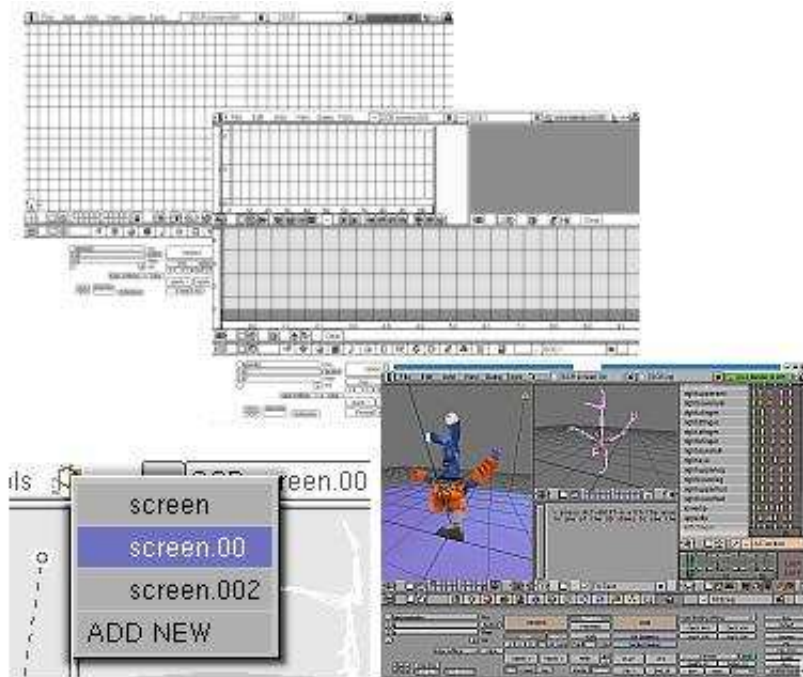
Sotto alla 3D Window, trovate la Button Window, come indica l'iconcina, nella stessa posizione delle altre, che rappresenta due bottoni.



## 2.3 Gli Screens e le Windows

Blender puo' contenere molti *Screens*. Ogni Screen si occupa di una combinazione di finestre diversa; questo e' indispensabile quando si lavora a livello professionale e si ha bisogno, come dire, di un tavolo da lavoro con piu' cassetti. Si puo' avere, ad esempio, uno screen con quattro finestre che mostrano l'oggetto da diverse angolazioni, uno screen con due finestre per l'editor sonoro e per le animazioni, uno screen con le finestre di gestione dei file, e cosi' via...

Lo Screen corrente viene indicato nella Info Window; accanto al suo nome vi sono i tasti per aggiungerne di nuovi, per cancellare quello corrente o per spostarvi da uno screen all'altro (per spostarvi da uno Screen all'altro, potete anche utilizzare la combinazione di tasti (in gergo si dice *Hot Keys*) *CTRL + SHIFT + NUMPAD 4* oppure *CTRL + SHIFT + NUMPAD 6*) Le 3 window che ci troviamo davanti sono le finestre predefinite che Blender ci mostra allorquando viene aperto; tutte insieme esse occupano lo Screen di default. Sempre per default vi sono memorizzati tre screen, chiamati *screen.001*, *screen*, *screen.002* : provate a vedere come sono organizzati.



Blender non e' certo limitato ad avere un numero massimo di finestre: per la verita', in teoria, esso permette di suddividere la schermata in infinite window, con la possibilita' di far assolvere ad ogni window un compito diverso.

Vediamo come si suddivide uno Screen in ulteriori finestre. Andiamo sul bordo della finestra da suddividere, e troviamo la posizione in cui il cursore cambia spontaneamente aspetto trasformandosi nella doppia freccia del trascinamento; adesso premiamo il tasto destro del mouse; comparira' una finestra Pop-Up: premendo su *Split Area*, potremo scegliere il punto dove dividere (splittare) la finestra maggiore. Dovendo dividere in due una finestra 3d che "confina" con un'altra finestra 3d, dovremo porre attenzione su quale delle due si affaccia una porzione maggiore di cursore: sara' questa

ad essere divisa. Sembra complicato a scriverlo, ma in realta' e' semplice ed intuitivo.



Allo stesso modo, se vogliamo cancellare una finestra che non ci interessa piu', basta premere su *Join Area*, che ci permettera' di unire le due finestre toccate dal cursore in quel momento. Se accettiamo, le due finestre verranno

fuse in una sola, e precisamente quella sulla quale vi era una porzione maggiore di cursore. Ad ogni finestra creata in questo modo possiamo dare una funzione diversa.

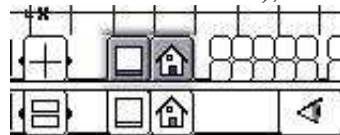
Come si fa? Ogni nuova window crea una propria header (per default nella sua zona inferiore), che come le altre, ha la propria iconcina a sinistra. Tale iconcina, come gia' detto, permette di capire e cambiare la funzione svolta dalla relativa finestra. Premendola comparira' una pop-up con tutti i tipi di finestre disponibili; sara' allora

sufficiente scegliere quella che si vuole.

Lo screen personalizzato potra' cosi' contenere tutti i tipi di finestre necessarie, ma solamente una window alla volta potra' essere attiva in un determinato momento, e solo questa potra' ricevere gli input dell'utente. Per lavorare sulla finestra desiderata, e' sufficiente trovarsi al di sopra di essa con il cursore: e' una regola semplicissima, ma bisogna fare molta attenzione su dove si sta lavorando, perche' le stesse combinazioni di tasti hanno effetti differenti a seconda del contesto. Quando vi spostate con il mouse sullo screen, la finestra corrente viene segnalata con lo schiarirsi del suo header. Per ridimensionare una window, muovete semplicemente il cursore del mouse sul margine della window.

Il cursore prendera' la famosa forma di trascinamento. Premete e mantenete premuto *LeftMouse* per muovere il margine a vostro piacimento...

All'interno di quasi tutte le window troviamo due tasti che servono anch'essi per la navigazione: una finestrella bianca ed una sorta di casetta. La prima icona serve a portare la finestra corrente nella modalita' Full Screen (Gli Hot Keys corrispondenti sono *CTRL + SHIFT + NUMPAD8* e *CTRL + SHIFT + NUMPAD2*), la seconda riporta la vista alle impostazioni iniziali.



Dopo aver creato una disposizione soddisfacente delle finestre la potete salvare come impostazione di default, premendo contemporaneamente *CTRL + U*. Questi sono i tipi di Windows attualmente supportati:

1. *Character Tools*, per animare i personaggi.
2. *Sounds Editor*, per montare i suoni delle sequenze.
3. *Image Viewer*, per la preview e per caricare i file immagine.
4. *Script editor*, per visualizzare, modificare ed utilizzare testi e python script
5. *SequenceWindow*. Per il video editing e la post produzione.
6. *Info Window*, per le impostazioni e le informazioni.
7. *Editor di immagini* per le mappature UV
8. *FileWindow*. Per caricare e salvare i files.
9. *ButtonsWindow*. La visualizzazione degli specifici block types.
10. *OopsWindow*. Lo schematico diagramma che mostra i data blocks.
11. *IpoWindow*. Per le curve di animazione ed i VertexKeys.
12. *3DWindow*. Mostra e modella gli Oggetti come wireframe, solidi o Gouraud shaded.

## 2.4 Elementi di partenza

### La 3D view, il mirino, il piano iniziale.

La finestra centrale, che come già accennato è la *3D view*, occupa per default la parte centrale dello screen. Il quadrato che si vede al centro è uno degli oggetti di Blender, il *piano standard*, che viene creato per default (all'apertura Blender carica il file *.B.blend*, che contiene questa configurazione. È questo il file che viene aggiornato quando si utilizza la combinazione *ctrl-U* per salvare le impostazioni personalizzate; a parte il nome, è un normale file di Blender) con lo scopo di fornire un orientamento di base, mentre il 'mirino' è il cursore 3d.

### Impostazione della 3D view

Cliccando sull'iconcina della 3D view, come già spiegato compare una pop-up contenente tutti i vari tipi di finestre. Andando a scegliere quella che rappresenta una coppia di bottoni, la finestra cambierà in Button window. Per default la sua header si troverà nel modo configura-vista, rappresentato dal disegno di occhio di profilo e conterrà una serie di bottoni per la configurazione della visualizzazione della *3D view*. Occorre notare che questi particolari bottoni non compaiono se si preme semplicemente l'iconcina ad occhio, ma è necessario passare dalla *3D view* alla *Buttonview*.

Nel gruppo di tasti così comparso ve n'è uno particolarmente importante, denominato *BackGround-Pic*. Premendolo sarà possibile scegliere uno sfondo da visualizzare nella finestra, molto utile, ad esempio, quando si vuol ricaricare un'immagine.

Non appena si schiaccia tale bottone, si aggiungono nuovi pulsanti, cioè uno che riporta il percorso utilizzato per trovare l'immagine, un altro che attiva una gestione dei file grafici con anteprima, uno con la percentuale di sovrapposizione dell'immagine rispetto alla griglia di riferimento (Blend) ed un altro che prevede l'utilizzo di una texture animata come immagine di sfondo.

### Il Pivot Point e la camera standard.

Notate che vi è anche una piccola sfera: è il baricentro del suddetto piano.

Qualsiasi trasformazione riguardante un oggetto viene riferita al proprio baricentro grafico (poiché il baricentro grafico può sempre essere riposizionato dall'utente, non sempre coincide con il baricentro fisico dell'oggetto...).

Il termine tecnico per indicare questo importante elemento è *Pivot Point*.

Diremo quindi che la piccola sfera è il *Pivot Point* del piano. Ogni oggetto ha un suo *Pivot Point*, attorno al quale viene ruotato e scalato. Per default ci troviamo nella vista dall'alto, o top view. Il triangolo in basso rappresenta il cono visivo della telecamera di default. Cliccando con il pulsante sinistro del mouse nella *3DWindow* il mirino si posiziona nel punto dove si trova il cursore. Il mirino ha la funzione di indicare dove si posizioneranno o dove verranno spostati i vari oggetti (con determinate opzioni, il mirino può sostituire il *Pivot Point* degli oggetti al momento della rotazione e della scalatura).

## 2.5 La modalita' Trackball: attorno all'oggetto.

La modalita' *Trackball* è lo stato in cui si trova la *3Dwindow* quando si naviga al suo interno. In pratica, nel momento in cui zoomiamo e giriamo intorno alla nostra scena, stiamo già utilizzando tale modalita'. Vi sono tre modi diversi di "navigare" in tale ambito.

- Tenendo premuto *Mmouse* oppure *ALT + Lmouse*, e muovendo contemporaneamente il cursore nella *3DWindow*, ruotate il vostro punto di vista (se ci pensate sembra proprio di usare una trackball). Se riprovate tenendo premuto il tasto *SHIFT + MiddleMouse* attivate la traslazione della *3Dwindow*. *CTRL + MiddleMouse* attiva invece lo zoom in e out.
- Utilizzando i tasti del gruppo numerico, detto PAD, accertatevi che Num Lock sulla vostra tastiera sia illuminato) potete forse controllare meglio la situazione. Premendo *PADPLUS* e *PADMINUS*) attivate lo zoom in e lo zoom out. Per la rotazione invece, dovete usare: *PAD2*

*PAD4, PAD6, PAD8* (notate le frecce dei tasti !). Per le traslazioni premete *SHIFT* mentre usate *PAD2 PAD4, PAD6, PAD8*.

- Le opzioni di traslazione e zoom sono disponibili anche come bottoni negli header di ogni *3DWindow*;

## 2.6 Le 3 viste classiche: sopra, davanti e di lato

Anche le viste standard *Top, Front* e *Side* sono raggiungibili tramite il tastierino numerico (notate la collocazione "logica" dei tasti sul PAD):

- Top view: *PAD7*.
- Front view: *PAD1*.
- Side view: *PAD3*.

Il tipo di vista puo' anche essere selezionato premendo l'apposito pulsante sulla *header* della *3DWindow*: la finestra pop up dara' la possibilita' di scegliere tra S (side), F (front) e T (top).

## 2.7 View modes

Mentre con i comandi appena descritti e' possibile cambiare solo il punto di vista, con quelli che stiamo per vedere possiamo invece modificare il "modo" in cui vediamo un oggetto. E precisamente abbiamo tre possibilita' principali:

- *Ortonormale*, e' il view mode di base per modellare.
- *Prospettica*. utilizzate *PAD5* per passare dal modo Ortonormale a quello Prospettico e viceversa.
- *Camera view*: utilizzate *PAD0*. Potete uscire dalla Camera view invocando qualsiasi comando view.

## 2.8 View Object

Esiste un ulteriore punto di vista, poco intuitivo ma molto utile: il *Pivot Point* degli oggetti. In Pratica qualunque oggetto puo' diventare una "telecamera", ed il rendering puo' essere effettuato dal punto di vista del suo *Pivot*.

Per default l'oggetto guarda verso il basso, ma bisogna fare attenzione al tipo di primitiva utilizzata: se si tratta di un piano o di una curva non c'e' problema, mentre se si utilizza un solido occorre fare attenzione al fatto che il Pivot si trova al suo interno, e quindi l'eventuale vista sara' bloccata dalla superficie stessa della primitiva, a meno che questa non sia di materiale trasparente oppure non si trovi su di un layer che non viene considerato al momento del rendering. Uno tra i tanti usi di questa capacita' e' la simulazione di uno specchio senza ricorrere alla mappatura ambiente: dopo aver fatto un rendering dal punto di vista dell'oggetto "specchio" si utilizza l'immagine ottenuta per mapparla, ottenendo un risultato molto convincente, e a volte migliore di quello realizzato con la mappatura ambiente.

Per settare un oggetto come telecamera, occorre, dopo averlo selezionato, premere *CTRL + Zerokey*, mentre per ripristinare la telecamera di default, basta usare i tasti *ALT + Zerokey*.

## 2.9 Layers

Ogni finestra puo' contenere dei *livelli*, o *layers*, fino ad un massimo di 20. I livelli servono a gestire la visibilita' degli oggetti, in quanto ogni layer puo' essere reso visibile o invisibile (con tutto quello che contiene) a piacimento.

Immaginiamo di avere una scena molto complessa, con molti elementi che si sovrappongono e che rendono difficile persino la selezione: potremmo raggruppare in un layer tutti gli attori, in un altro tutti i mobili, e in altro ancora gli sfondi e le pareti.

Potremo cosi' decidere se vedere tutto o solo una parte, alleggerendo il lavoro anche al computer stesso. I *layer* vengono visualizzati nella *header* della *3dWindow*, sotto forma di pulsantiera. Il primo pulsante indica il livello 1, il secondo il livello 2, il terzo il livello 3 etc...

Quando un pulsante e' visualizzato come "premuta" cio' indica che il layer corrispondente e' attivo. Per abbassare piu' pulsanti contemporaneamente (quindi per attivare piu' layer insieme) occorre premerli con *Rmouse* + *SHIFT*.

Per attribuire un oggetto ad un determinato layer, occorre, dopo aver selezionato l'oggetto, premere il tasto *Mkey* : comparira' una finestra pop up che riproduce la pulsantiera dei layer; premendo il tasto corrispondente al layer che vogliamo assegnare all'oggetto completeremo l'operazione.

Possiamo anche premere solo il numero da 1 a 20 corrispondente al layer che desideriamo assegnare.

## 2.10 Selezioni

Questi sono i tre metodi per selezionare:

- Utilizzando *RightMouse* si seleziona (e si attiva) sempre un singolo elemento (item), tutto il resto viene deselezionato. Comunque , tenendo premuto *SHIFT* durante la selezione si estende la selezione.
- Gli *Items* possono essere selezionati anche tramite un *box* di selezione:premete *Bkey* (sta per *Border*) e disegnate un rettangolo con *LeftMouse* per selezionare, *RightMouse* per deselezionare.
- La hotkey *Akey* (All ) e' un toggle per selezionare o per deselezionare tutti gli items.

Vi sono altre caratteristiche riguardo ai vari tipi di selezione, ma sono piu' inerenti all'*edit mode*, che vedremo tra breve.

## 2.11 Oggetti attivi e oggetti selezionati.

Blender fa distinzione tra selezionato ed attivo.

- Solo un oggetto puo' essere *attivo* in un determinato momento. Questo e' necessario, ad esempio per permettere la visualizzazione dei dati nei buttons. l' oggetto attivo e selezionato e' mostrato con un colore piu' chiaro degli altri oggetti selezionati.
- Un numero qualsiasi di oggetti puo' essere *selezionato* in una volta sola. Virtualmente tutti i comandi attivabili da tastiera hanno effetto sugli oggetti selezionati.

Questa e' una distinzione importante a causa dell'interfaccia di Blender che non pone blocchi all'utente. La maggior parte delle azioni attivabili da *hot key* operano su tutti i dati selezionati. Ma la *ButtonsWindow* e la *IpoWindow* possono mostrare il contenuto solo dei dati attivi. Per fare un esempio: posso selezionare quanti oggetti voglio, ma nel material button, vedro' solo il materiale dell'entita' attiva.

---

## 3. Editing di base

2002-11-23 16:57:24

Contenuto della sezione

### 3.1 Le primitive

Cliccando sul tasto in alto a sinistra immediatamente riconoscibile grazie all'icona di punto interrogativo oppure premendo la *SPACEBAR*, compare il menu' principale di Blender, che fino alla versione 1.80 era anche l'unica interfaccia per accedere ai comandi sui file.

Dalla versione 2.12 si puo' accedere anche tramite il menu' a tendina come i normali programmi windows.

Le primitive sono chiamate in questo modo perche' si tratta delle figure base dalle quali e' possibile ottenere, mediante le operazioni di modellazione, qualsiasi altro oggetto complesso.

La primitiva per eccellenza e' il *piano*, un oggetto bidimensionale formato da una sola faccia e quattro vertici. Si tratta della figura di default che troviamo all'apertura di Blender.

### 3.2 Mesh

Le mesh sono oggetti tridimensionali formati da centinaia di poligoni connessi tra loro. Una superficie sferica, ad esempio, potrebbe essere rappresentata da una moltitudine di piccoli triangoli. Ovviamente, maggiore e' il numero di poligoni utilizzato, migliore e' l'approssimazione della superficie. Ogni triangolo possiede un dritto ed un rovescio, o meglio un interno ed un esterno. Quando il computer calcola l'aspetto di un oggetto, si comporta diversamente se la luce sbatte sulla superficie esterna o su quella interna.

Spesso una superficie rovesciata non viene visualizzata affatto, poiche' Blender presume che le facce interne non debbano essere calcolate durante il rendering. Se un lato sia il "dentro" o "fuori" di un oggetto viene determinato dai *vettori normali*, che come vedremo piu' tardi possono essere visualizzati e modificati in fase di editing. Vediamo ora i vari tipi di mesh forniti da Blender.

### 3.3 Le Metaballs

Le metaballs, o blob, possono essere pensate come sfere di metallo liquido magnetizzate, in grado cioe' di attrarsi o respingersi secondo particolari criteri di forza. D'altronde la loro ideazione pare sia dovuta ai modelli molecolari usati in chimica. La creazione della metaballs e' facilissima, basta sceglierle, come al solito dal menu' *ADDMetaball*. Attenzione, appena creato, il blob si trova in stato di editmode, ed occorre premere *TAB* per uscirne.

l'utilizzo principale che se ne fa e' quello di modellare figure organiche, dotate di muscoli o comunque nodose e morbide, ad esempio il fusto di un ulivo, o il corpo di un supereroe. l'apparenza di una metaball singola e' quella di un nucleo di materia sferoidale avvolta da un campo di forza magnetico, rappresentato, quest'ultimo, da un cerchio. E' bene sapere che benche' in teoria si possono utilizzare contemporaneamente tutte le metaballs che vo-gliamo, esse formano comunque un solo oggetto. Ce ne accorgiamo in due modi:

1. Quando decidiamo di convertire anche un solo blob in mesh, in modo da poterlo lavorare con le tecniche usuali, osserviamo che la conversione

avviene per tutte i blob creati.

2. I parametri metaball appaiono solo se selezioniamo il primo blob creato; tutti gli altri sono suoi 'figli' e ne ereditano ogni caratteristica. Modificando la metaball di partenza, si modifica l'intero oggetto globale.

A livello oggetto noi possiamo eseguire tutte le normali operazioni di rotazione, scalatura e traslazione, nonché, grazie agli Edit-Button, decidere la risoluzione delle metaballs in fase di lavorazione (Spesso è utile lavorare a bassa risoluzione, per non impegnare troppo il computer, per

poi fare un rendering ad alta risoluzione) ed in fase di rendering (bottoni tipo slice wiresize e rendersize). Per alleggerire ulteriormente la lavorazione, Blender offre, sempre a livello oggetto e sempre con gli Edit-Button, la possibilità di decidere in che modo gestire i complessi calcoli di forma che avvengono quando una delle metaballs viene spostata. l'opzione di default è quella che aggiorna sempre ed in real time le trasformazioni.

Ma volendo evitare questo lavoro al nostro processore potremmo decidere di vedere i blob, durante gli spostamenti, a mezza risoluzione (rispetto a quella scelta) o addirittura di

vederne solo il campo di forza (bottoni update: always, half res, fast). Molte altre importanti caratteristiche, come la forma, i parametri di attrazione etc. . . sono accessibili solo selezionando la prima balls ed entrando in Edit-Mode. Vedremo queste specifiche tecniche nel capitolo sulla modellazione avanzata.

## 3.4 Le NURBS

Le NURBS, non-uniform rational B-spline sono curve e superfici ottenute grazie ad un particolare algoritmo matematico che permette di descriverne la topologia utilizzando pochi punti di controllo, e garantendo sempre la formazione di solidi morbidi e ben raccordati. Blender ne permette la creazione come al solito premendo la barra spazio, e poi ADD —>SURFACES. Di seguito è possibile scegliere tra diverse nurbs di base:

Curve

Circle

Surfaces

Tube

Sphere

Donut

La differenza tra questi oggetti e quelli creati con la tecnologia delle mesh si vede subito dal fatto che in Edit-mode compaiono pochi punti di controllo, e che non occorre impostare le suddivisioni U e V (come accade ad esempio nella sfera mesh), poiché si ottiene immediatamente una superficie liscia. Se poi ci divertiamo a tirare qualche punto di controllo ci accorgeremo che la geometria delle superfici si deforma in modo armonioso, senza creare brutte fratture o spigoli, come nel caso delle mesh.

## 3.5 Trasformazioni di base: muovere, ruotare,

Molte azioni in Blender riguardano il muovere, ruotare o cambiare la grandezza di certi items. Si tratta delle operazioni di base, che possono essere applicate a qualsiasi oggetto o sub-oggetto. Possiamo quindi ruotare, traslare e scalare sia una sfera, sia una porzione dei suoi punti, oppure un curva di Bezier, o solo una delle sue maniglie di controllo. Blender offre un grande range di opzioni per ottenere il risultato che vi siete prefissi. Le stesse operazioni possono essere compiute anche sulle curve della ipowindow, per modificare la velocità e l'ampiezza delle animazioni.

## 3.6 Traslazione

Prestando 'G' (Grab mode), entriamo in modo traslazione. Muovendo il mouse spostare l'item selezionato, e dopo averlo portato nella posizione desiderata, lo 'fissiamo' premendo LeftMouse o ENTER o SPACE per assegnare la nuova locazione. Premendo ESC annulliamo l'operazione e l'ente ritorna al suo punto di partenza. Il movimento è sempre corretto per il tipo di view nella 3Dwindow. Per corretto, intendiamo che si svolge su di un piano normale all'asse visivo della veduta selezionata: La vista Top trasla sugli assi x,y, tra l'altro indicati in basso a sinistra. La vista Side trasla sugli assi y,z. La vista Front trasla sugli assi x,z. Possiamo usare MiddleMouse come un toggle per limitare la traslazione agli assi X, Y o Z. Blender determina quale asse usare, basandosi sul movimento già iniziato dal mouse.

RightMouse, tieni-muovi. Se dopo aver selezionato un oggetto con questo pulsante, continuiamo a tenerlo premuto, entriamo automaticamente in Grab mode.

### 3.4.2 Rotazione

Prestando 'R' entriamo in Rotation mode. Possiamo ruotare l'ente selezionato attorno al centro di rotazione prestabilito. Muovi il mouse attorno al centro di rotazione, premi LeftMouse o ENTER o SPACE per assegnare la rotazione. Basta premere ESC per annullare. La Rotazione è sempre perpendicolare alla view della 3DWindow.

## 3.7 Rotazione

Prestando 'R' entriamo in Rotation mode. Possiamo ruotare l'ente selezionato attorno al centro di rotazione prestabilito. Muovi il mouse attorno al centro di rotazione, premi LeftMouse o ENTER o SPACE per assegnare la rotazione. Basta premere ESC per annullare. La Rotazione è sempre perpendicolare alla view della 3DWindow.

## 3.8 Dimensioni (Scaling)

SKEY, Scaling mode. Muovi il mouse dal centro di rotazione verso l'esterno, premi LeftMouse o ENTER o SPACE per assegnare il ridimensionamento. Usa MiddleMouse come un toggle per limitare il ridimensionamento rispetto agli assi X, Y o Z. Blender determina l'asse appropriato basandosi sulla direzione del movimento del mouse.

## 3.9 Trasformazioni attivate dai movimenti del mouse

Tenendo premuto il pulsante sinistro del mouse, e tracciando dei segni specifici, Blender esegue gli stessi comandi descritti sopra:  
Tracciando una linea retta (all'incirca) si attiva la modalita' di traslazione.  
Tracciando un cerchio (all'incirca) si attiva la modalita' di rotazione  
Tracciando una 'V' (all'incirca) si attiva la modalita' di Scaling.

## 3.10 Snap

Lo snap rappresenta la possibilita' di muoversi con estrema precisione all'interno della scena disegnata. Blender supporta due tipi di snap: a cursore e a griglia.

## 3.11 Lo snap a griglia

E' possibile spostare, ruotare o ridimensionare gli oggetti saltando in maniera ortogonale ad intervalli pari al lato dei quadrati della griglia, senza il rischio di fermarsi nei punti intermedi. Quindi, se ad esempio la griglia e' suddivisa in tanti quadrati di 0.5 unita' di lato, potremo ottenere con facilità segmenti di 0.5, 1, 1.5, 2, etc. . . La precisione con cui vogliamo snappare dipende essenzialmente dal valore della densita' della griglia.

## 3.12 Settaggio della griglia e snap cursore

Abbiamo già parlato a pag. 14 della parte nascosta della Info Window, che viene resa visibile trascinando in basso il bordo inferiore della sua header. Se osserviamo bene, tra questi pulsanti ce ne sono tre che interessano lo snap: uno più grande, Grab Grid e sotto di esso altri due Size e Rot. Premendo il primo si attiva la modalita' per cui l'operazione di traslazione avviene saltando in base alla densita' della griglia. La rotazione e la scalatura rimangono però libere, a meno che non vengano premuti gli altri due pulsanti. Lo stesso effetto di questi pulsanti si ottiene temporaneamente premendo il tasto CTRL durante le varie trasformazioni. Questo permette di ottenere trasformazioni di precisione, ruotando ad esempio ad intervalli precisi di 5 gradi (con le impostazioni di default). Ma come settare l'intervallo della griglia? Ricordiamoci del settaggio della 3D Window descritto a pag. 17. Tra gli altri bottoni ne compare uno chiamato Grid. Il suo valore corrisponde alla distanza tra due nodi. Esiste però un problema con questo genere di snap: se il punto di partenza non coincide con un nodo, poiché lo spostamento sarà regolare, anche il

punto finale non coincidera' con un nodo. Per ovviare a questo inconveniente, e' opportuno portare prima il Pivot Point dell'oggetto sopra un nodo, in modo da avere un riferimento regolare. Questo si ottiene con la combinazione 'SHIFT S' (ovviamente il cursore deve trovarsi sopra la 3D Window) che visualizzera' una pop up contenente il seguente indice:

Sel -> Grid

Sel -> Curs

Curs -> Grid

Curs -> Sel

La prima opzione sposterà il Pivot Point dell'oggetto selezionato nel nodo più vicino. La seconda lo porterà invece dove si trova il cursore. La terza e la quarta portano il cursore rispettivamente sul nodo più vicino o nel punto medio della selezione. Gli spostamenti legati al cursore rappresentano lo snap cursore. In fase di edit mode, come vedremo, questa caratteristica è importantissima.

---

## 4. Material editing

2003-01-03 22:39:12

Contenuto della sezione

### 4.1 UV mapping

La mappatura UV permette di posizionare un'immagine su di un oggetto (oppure su una o più facce del medesimo) con estrema precisione.

Occorre dire che questa tecnica è strettamente legata alle librerie OpenGL, essendo anche usata per la visualizzazione in real time all'interno del game engine.

Per tale motivo, le dimensioni in pixel delle texture usate dovranno essere multipli e sottomultipli di 256, e dovranno essere quadrate. Ad esempio potranno essere utilizzate immagini di 8x8, 16x16, 32x32, 64x64, 128x128, 256x256. . . Per utilizzare questa tecnica occorre servirsi innanzitutto di due finestre, oltre a quella dei pulsanti: la 3D windows e la Image Windows.

[IMAGE]

Image window prima del caricamento di una qualsiasi immagine.

Andando con il mouse sopra la prima, dopo aver selezionato l'oggetto da mappare, e premendo FKEY entriamo in modalità face select. Otterremo lo stesso risultato spingendo il tastino a forma di triangolo, nella header della 3D Windows.

[IMAGE]

Fatto ciò osserveremo che l'oggetto assume una nuova ombreggiatura, e ci permette di selezionare le proprie facce cliccandoci sopra con RIGHTMOUSE. Tenendo premuto SHIFT, al solito, potremo selezionare più facce, contemporaneamente.

### 4.2 Caricamento della bitmap

Ora non ci rimane altro che caricare la bitmap scelta, che verrà assegnata solo alle facce selezionate. Per effettuare questa operazione basta premere il tasto LOAD, che si trova nella header della Image window. La finestra di gestione file che comparirà a posto della Image Window sarà munita anche di un'ottima preview delle immagini compatibili (.tga o .jpg). Passandoci sopra con il mouse, in basso verranno fornite anche alcune utili informazioni, come il

peso in KB, le dimensioni in pixel, il formato ed il tipo di compressione usato. Se e' attivata l'icona delle lente di ingrandimento, sempre nella header, il passaggio su di una immagine dara' luogo anche ad un moderato zoom della preview.

Dopo aver selezionato la thumbnail 1 sara' sufficiente premere il pulsante della tastiera INVIO. Si tornera' in questo modo alla Image window, solo che adesso sara' presente la bitmap caricata, che sara' gia' stata automaticamente assegnata alle facce selezionate. Se proviamo, nella 3D window, a selezionare nuove facce, l'immagine scomparira' dalla Image window, poiche' tale facce non possiedono ancora nessuna bitmap. Per assegnare anche ad esse la stessa texture, non occorrera' caricarla di nuovo con il tasto LOAD, ma sara' sufficiente selezionarla tra quelle gia' disponibili, premendo il tastino con una sorta di lineetta orizzontale. La procedura LOAD occorre solo per caricare nuove texture.

Mentre nella 3D window selezioniamo le facce, osserviamo che contemporaneamente nella Image window si evidenziano delle aree.

Si tratta delle aree di texture corrispondenti alle facce selezionate.

In pratica, ad ogni faccia della mesh corrisponde una zona dell'Image window, secondo determinati tipi di proiezione.

### 4.3 Tipi di proiezione UV

Il problema che nasce nella definizione di un materiale riguarda la difficolta' di usare immagini bidimensionali, di forma quadrata o rettangolare, per 'avvolgere' o 'rivestire' oggetti 3D di forma molto complessa. Ovviamente, questo dara' luogo a delle deformazioni, che occorre controllare, se non addirittura volgere al proprio scopo.

In ogni caso si tratta di una proiezione, o mapping dell'oggetto, che puo' avvenire in base a diversi algoritmi. Teniamo presente che Blender permette di assegnare diversi tipi di mapping a diverse facce dello stesso oggetto.

Andando con il mouse sulla 3D window, dopo essersi assicurati di essere in face select mode premendo UKEY si accede alla finestra pop up UV CALCULATION che ci permette di scegliere i vari tipi di mapping. Questi sono:

Cube

Cylinder

Sphere

Bounds to 1/4

Bounds to 1/2

Standard 1/4

Standard 1/2

Standard 1/1

From Window

l'ultimo tipo di proiezione e' quello piu' facile da prevedere, poiche' proietta la bitmap in modo tale che essa rimanga invariata addosso all'oggetto visto nell'angolazione corrente della 3D window. E' interessante notare che nella Image window le facce selezionate vengano riprodotte distorte in base alla proiezione scelta. Così scegliendo il calcolo UV di tipo cylinder, esse verranno srotolate attorno all'asse verticale, mentre con il tipo sphere si otterra' uno sviluppo sferico.

## 4.4 UV editing

Da fare...

## 4.5 Rendering di mappe UV

Spesso sono sorte delle perplessità riguardo al corretto modo di usare le mappe posizionate con coordinate UV all'interno del motore di rendering.

Ecco alcune brevi note.

Dopo aver UVmappato un oggetto con l'immagine "pippo.tga" se tale oggetto ha un materiale semplice, il rendering ignorerà la texture e le sue coordinate.

Premendo "texface" invece il rendering userà correttamente la UVmap, senza neppure il bisogno di settare "UV" come sistema di mappatura.

Solo che in questo modo è impossibile usare la potenza dei materiali di Blender: niente bump, alpha o altro...solo il colore fornito da pippo.tga in base alle coordinate UV.

Cosa è possibile fare allora?

Aggiungere nel solito modo la texture al materiale, avendo cura di utilizzare lo stesso pippo.tga

Ora pippo.tga compare anche nel material editor. Adesso si che bisogna settare "UV" come sistema di coordinate: in tal modo la texture appena caricata e facente parte del vero materiale, userà le coordinate settate precedentemente e verrà renderizzata in base ad esse.

Usando tale "vero materiale", si può tranquillamente disattivare "texface": ora non ci serve più.

## 4.6 Effetti speciali

Utilizzando le caratteristiche dei materiali di Blender, è possibile ottenere degli effetti "non fotorealistici", come l'effetto cartoon, wire solid, etc..

## 4.7 Sfumatura wire-solid

Interessanti effetti possono essere ottenuti utilizzando due oggetti identici sovrapposti, mappandone uno con una texture in Ztransp e l'altro con una texture tipo Wire.

In questo modo si ottengono le immagini molto suggestive e diffusamente impiegate, di un oggetto meta' reale e meta' virtuale...

---

## 5. Editing avanzato

2002-12-11 23:36:21

Contenuto della sezione

## 5.1 Oggetti ed EditMode

Ogni oggetto e' formato da altri oggetti piu' piccoli. Pensiamo ad un edificio che a sua volta e' composto da mattoni, porte, piastrelle etc. . .

Quando noi lavoriamo a livello di oggetto e' come se intendessimo lavorare su tutta la casa intesa come un'unica entita', per cui la ruotiamo tutta, la scaliamo tutta, la spostiamo tutta e la cancelliamo tutta.

Lavorare a livello di sotto-oggetto invece potrebbe essere inteso come lavorare sui singoli mattoni, cancellare una finestra, spostare un gruppo di mattonelle etc. . .

Nel caso specifico della grafica 3D ogni entita' possiede una propria modalita' sotto-oggetto; ad esempio una sfera mesh viene scalata, spostata e ruotata a livello di oggetto, mentre volendo agire solo sulle sue facce o sui suoi vertici occorre scendere a livello di sotto-oggetto.

Al contrario nel livello sotto-oggetto di una sfera blob non esistono vertici o facce, ma altri parametri specifici. Dopo aver selezionato l'oggetto da editare, si accede alle proprieta' sub-object in tre modi:

Premere ALT-E

Premere TAB

Nel frame della 3d window, identificato, come gia' detto dall'icona della finestra divisa in quattro parti, premere il pulsante a forma di piccolo triangolo.

All'interno della modalita' sub-object l'editing e' bloccato allo specifico Ob-Data block dell'Oggetto attivo, per cui non e' piu' possibile selezionare altre entita' esterne, ma si puo' solo lavorare sui sotto oggetti dell'entita' selezionata. Tutta una nuova serie di tools si rende ora disponibile.

Potrete aggiungere vertici e facce, estrarre poligoni, disegnare curve o assegnare colori.

Per tornare in modalita' object si ripete la stessa operazione fatta per entrarvi. Finche' non uscirete dall'edit mode potrete utilizzare la U-KEY che qui ha la funzione di Undo, e ripristina i dati prima dell'ingresso a livello sub-object.

Per ricordarvi che siete in EditMode, il cursore del mouse cambia prendendo la forma di una croce. Vedremo poi in seguito le possibilita' di questa modalita'.

## 5.2 Shrink/Fatten (ALT-S)

Da fare...

## 5.3 Il pulsante Hash

I vertici degli oggetti mesh sono memorizzati all'interno di Blender secondo un certo ordine. Normalmente questo ordine non e' importante ai fini del risultato, poiche' l'importante e' che le coordinate spaziali (x,y,z) di ogni vertice rimangano immutate. Tuttavia vi sono alcuni casi in cui in cui questo fattore diventa importante, ad esempio nel caso in cui un python script debba recuperare le posizioni dei vertici per poterle rielaborare, oppure nel caso in cui l'oggetto in questione venga utilizzato come emettitore di particelle. Nell'ultimo caso e' bene sapere che l'ordine di emissione e' basato sull'ordine dei vertici. Il pulsante Hash mischia l'ordine dei vertici all'interno del database della scena, per cui un'eventuale emissione particellare diventa molto piu' casuale.

## 5.4 Il pulsante Xsort

Il pulsante Xsort e' praticamente l'inverso del pulsante Hash, poiche' ristabilisce l'ordine dei vertici in base alla loro posizione. La priorita' viene stabilita andando da sinistra a destra, da sopra a sotto.

## 5.5 Creazione di facce

Per creare una singola faccia occorrono almeno tre punti (vertex)...da fare

## 5.6 Operazioni booleane

La modellazione booleana permette di aggiungere, sottrarre od intersecare oggetti mesh. Per la verita' Blender non calcola addizione e sottrazione, ma solo l'intersezione, dalla quale poi le altre due operazioni derivano. La sequenza e' piuttosto semplice.

Si selezionano tutti gli oggetti che si intendono modellare (Rightmouse + Shift) .

Si uniscono tutti in un unico oggetto (CTRL+J)

Si entra in modalita' Edit-mode (TAB).

Si selezionano tutti i vertici (AKEY)

Si visualizza la bottoniera degli Edit-button (F9) e si preme il tasto Inter-sect.

L'oggetto viene scomposto in piu' parti, a seconda del numero di superfici che si intersecano. A questo punto, sempre all'interno di Edit mode, si de-selezionano tutti i vertici e si selezionano solo quelli delle varie parti cosi'

formate. Per fare questo occorre accompagnare la semplice selezione con il tasto destro alla pressione del tasto 'I', che seleziona tutti i vertici collegati tra loro senza nessuna interruzione. Sempre a livello Edit-mode le varie parti possono cosi' essere spostate o duplicate. Tornando in modalita' oggetto il risultato e' un'unica entita'.

## 5.7 Lavorare con le NURBS

La potenza delle NURBS risiede soprattutto nella grande capacita' di modellazione di forme organiche mediante la tecnica del lofting, che riesce ad ottenere una superficie, detta skin, partendo da una serie di sezioni. Si pensi alla comodita' di questa tecnica nel dover modellare, ad esempio, una nave.

Poiche' predisposte per la generazione di superfici, le curve NURBS non si trovano, nel menu' generale, sotto la voce "Curve", bensì sotto la voce "Surf" .

Dunque, mediante

ADD->SURF->

Si accede ad un elenco di oggetti NURBS, tutti adatti alle tecniche che poi descriveremo.

Le primitive NURBS

attualmente supportate sono:

Curve

[IMAGE]

Circle

[IMAGE]

Surface

[IMAGE]

Tube

[IMAGE]

Sphere

[IMAGE]

Donut

[IMAGE]

---

## 6. Animazione

2002-08-24 12:43:15

Contenuto della sezione

### 6.1 Animazione di base: keyframes

Sostanzialmente la tecnica dei keyframes e' semplicissima: si tratta di dire a Blender che posizione deve avere un determinato oggetto in un determinato fotogramma del nostro filmato. Non dovremo certo stare a settare tutti i fotogrammi: se tra due posizioni intercorre un certo numero di keyframes, Blender si occuperà di eseguire un'interpolazione per rendere il movimento fluido e realistico.

### 6.2 Navigazione tra i frames

La navigazione attraverso i frames, ossia i fotogrammi, pu' avvenire in diversi modi. Per ora ci occuperemo del modo piu' semplice, cioe' quello che si serve del pulsante che riporta il frame corrente (quello vediamo in quel preciso momento). Vedremo sistemi piu' complessi, ma piu' efficaci, quando parleremo della IPO Window.

Il pulsante che ci interessa in questo momento si trova a destra in alto nel-la Button Window, e per default riporta il numero 1. Infatti anche una scena statica viene implicitamente trattata come un fotogramma, e precisamente il numero 1. Premendo la combinazione 'SHIFT LEFTARROW' si torna al primo fotogramma, mentre con 'SHIFT RIGHTARROW' si va all'ultimo. Premendo il pulsante un po' piu' a sinistra il numero di fotogramma decresce di una unita', premendolo un po' piu' a destra il numero aumenta, premendolo contemporaneamente con il tasto SHIFT possiamo digitare il numero desiderato (Questa caratteristica appartiene a tutti i tasti numerici di Blender, persino quelli generati con il python) . Lo stesso effetto pu' essere ottenuto semplicemente premendo i tasti 'LEFTARROW' per tornare indietro di un frame e 'RIGHTARROW' per andare avanti di un frame. Utilizzando invece i tasti 'UPARROW' e 'DOWNARROW' ci si sposta nei frame di 10 in 10. Andando con il mouse sopra la 3D Windows e premendo la combinazione 'ALT A', l'animazione parte in real time. Premere ESC per interromperla. NUMPAD9 causa il refresh dell'animazione.

## 6.3 Utilizzo della cinematica inversa

Un braccio umano pu' o raggiungere una determinata posizione in due modi: ruotando prima la spalla, poi il gomito, poi in polso ed infine il palmo. Prendendo il palmo e portandolo dove deve stare, lasciando che tutto il resto si muova di conseguenza in base a determinate regole cinematiche. Il primo metodo utilizza una sistemazione artigianale, che potremmo de-finire diretta, mentre il secondo fa uso della cinematica inversa, o IK (Inverse Kinematic).

## 6.4 Lo scheletro di IKA

In Blender, esiste un oggetto particolare che utilizza un complesso sistema di equazioni in grado di simulare la cinematica inversa. E' sostanzialmente una struttura formata da elementi rigidi uniti da giunti snodati. Tali elementi possono essere uniti in modo da formare veri e propri scheletri, di esseri umani, certamente, ma anche di qualsiasi altro elemento snodato, come una catena, un treno, un uccello in volo. . .

## 6.5 La IPO windows

Discorso introduttivo sulla IPO...

## 6.6 Animazione ciclica

Per animazione ciclica si intende la realizzazione di quegli effetti che si ripeto-  
no all'infinito, come ad esempio l'oscillazione di un pendolo, o la rotazione  
di un elica. Blender possiede dei tasti adibiti esclusivamente all'estensione  
all'infinito di un qualsiasi movimento, che sostanzialmente pu' o avvenire in  
due modi:

Estrapolando il movimento all'infinito;

Ripetendolo uguale a se stesso all'infinito.

Nel primo caso la direzione e la velocita' vengono interpolate, per cui si  
mantengono per tutta la durata dell'animazione. Puo' essere il caso di un  
missile lanciato nello spazio, o di un oggetto in caduta libera. Nel secondo  
caso il movimento viene ripetuto infinite volte, ricominciando sempre da ca-po.  
Per cui, dopo aver realizzato un modellino della Terra e della Luna, e  
dopo aver costruito un'animazione che faccia ruotare la Luna di 360 gradi  
attorno alla Terra, si pu' o ripetere questo movimento all'infinito. Per realiz-  
zare questo effetto ci sono quattro pulsanti nella IPO Windows: i primi due  
sono iconizzati da delle frecce, mentre i rimanenti sono rappresentati da  
una sorta di linea oscillante. Le funzioni sono rispettivamente:

Prolungamento all'infinito utilizzando l'ultima posizione come costante;

Prolungamento all'infinito interpolando la traiettoria;

Prolungamento ciclico;

Prolungamento ciclico interpolato.

l'utilizzo e' semplice: dopo aver selezionato la curva IPO che si desidera  
estendere all'infinito, basta premere il tasto corrispondente alla funzione  
desiderata. Il grafico si modificera' di conseguenza.

## 6.7 RVK (Relative vertex key)

Le tecniche tradizionali di animazione delle mesh prevedevano delle trasformazioni univoche: si preparava un oggetto A (ad esempio una testa con le labbra chiuse) e se ne faceva una copia B, che veniva modificata facendo però attenzione a non modificarne il numero di vertici (ad esempio si aprivano le labbra); era poi possibile, durante l'animazione, trasformare gradualmente l'oggetto (morphing) A nell'oggetto B (ottenendo l'apertura o la chiusura della bocca). Con questo sistema, per ottenere ad esempio, sette animazioni differenti, ossia

Faccia che ride

Faccia che ride con occhio destro chiuso

Faccia che ride con occhio sinistro chiuso

Faccia che ride con entrambi gli occhi chiusi

Faccia seria con occhio destro chiuso

Faccia seria con occhio sinistro chiuso

Faccia seria con entrambi gli occhi chiusi

Occorre fare sette oggetti finali, scegliendoli di volta in volta come obiettivo (morphing target) delle trasformazioni di una testa A (seria con gli occhi aperti). Con l'utilizzo delle RVK la cosa diventa molto più semplice: innanzitutto non occorre fare target, ma si lavora sempre sullo stesso oggetto.

Quello che si va a modificare sono delle istanze di quest'ultimo, chiamate key. Inoltre sono sufficienti solo tre di queste key:

Bocca che ride

Occhio destro chiuso

Occhio sinistro chiuso

tramite tre curve di controllo potremo poi miscelare i tre effetti sulla testa iniziale, in modo da ottenere tutte le combinazioni possibili.

## 6.8 Uso delle RVK

Prima di iniziare occorre aprire da un lato la IPO window, impostarla su RVK mode con l'apposito pulsante e spostare la Button window su Animation mode. Tra bottoni di quest'ultima, dovremo poi attivare le RVK premendo il pulsante Relative Keys. Quindi, dopo aver selezionato l'oggetto da animare, si comincia con il creare la posizione di base, ossia la basekey. E' molto semplice:

Nella 3D window, premere IKEY;

dalla popup che appare scegliere Mesh

Nella IPO window, compaiono immediatamente due curve: una orizzontale (gialla se selezionata, arancione se non selezionata) ed una in pendenza, rossa. Quest'ultima serve per la velocità, ma per le RVK non ci serve e può essere cancellata. Per farlo basta premere CANC dopo averla selezionata con il RMouse.

La linea orizzontale è la basekey. Quando la selezioneremo, l'oggetto tornerà allo stato in cui si trovava al momento in cui abbiamo premuto IKEY.

Ripetendo i due passaggi descritti sopra, possiamo creare decine di key per ogni oggetto. La basekey sarà arancione, tutte le altre saranno azzurre.

Nella IPO window, tutte le curve indicanti le varie key si sovrapporranno nascondendosi a vicenda. Per ovviare a questo piccolo inconveniente è sufficiente

selezionare di volta in volta le varie linee (con RMouse) e spostarle premendo GKEY (grabmode). Cominciamo infatti con lo spostare un po' piu' in basso la basekey: l'altezza di queste linee non ha alcuna importanza. Supponiamo ora di aver creato tre key: una di base, la cui linea di riferimento si trova sotto a tutte, e due di trasformazione. A questo punto le key esistono, ma hanno tutte lo stesso aspetto. Dobbiamo modellare le varie trasformazioni. Anche questo e' molto semplice: Nella IPO window selezioniamo la key da editare  
Nella 3D window andiamo in Edit mode premendo TAB  
Modelliamo i vertici a piacimento  
Usciamo da Editmode (Se provassimo a selezionare un'altra key mentre siamo ancora in Edit mode, Blender ci chiederebbe: 'Copy key after edit mode?' In caso di risposta affermativa l'aspetto della key appena selezionata verrebbe sostituita dalla nuova configurazione che stiamo modellando.).  
La key selezionata si e' gia' aggiornata automaticamente. Selezionando, ad esempio, la basekey e poi nuovo la key appena modificata, potremo valutare la riuscita della modellazione.

## 6.9 Sincronizzazione e morphing delle RVK

Una volta create le varie key non ci resta che gestire il modo in cui dovranno sovrapporsi durante l'animazione. Tornando all'esempio della testa, potrebbe servire che, al 50esimo fotogramma, la bocca sia aperta per il 20%, l'occhio destro sia chiuso per il 60% mentre quello sinistro sia completamente aperto. A sul bordo destro della IPO window, abbiamo un elenco di key: key1, key2, . . . key18. Selezionandole con LMouse, queste scritte cambiano colore, diventando bianche. Esse corrispondono alla distribuzione durante l'animazione delle key che abbiamo creato; cliccandoci sopra non compare nulla semplicemente perche' non abbiamo ancora impostato tale distribuzione. Blender associa in automatico il keynumber con l'ordine in cui abbiamo creato le varie key: la prima key creata sara' dunque chiamata key1, la seconda key2 e cosi' via. . .

Ora poniamo che la key creata per prima, dunque la key1, corrisponda all'apertura della bocca. Per gestirne il movimento dobbiamo fare le seguenti operazioni:

Con Lmouse selezioniamo key1

Tenendo premuto CTRL, con LMouse, nella IPO windows, disegniamo la curva di Bezier, tenendo presente che l'ascissa rappresenta i fotogrammi e l'ordinata la percentuale trasformazione (1 corrisponde al 100%, mentre valori superiori vengono estrapolati).

Muovendo l'indice di animazione (la barra verde verticale)(O utilizzando i tasti arrow) vedremo l'oggetto

trasformarsi nel tempo secondo i valori della curva di Bezier. Volendo modificare quest'ultima, dopo averla selezionata con RMouse, potremo entrare in edit mode con il tasto TAB, come se si trattasse di un qualsiasi oggetto 3D, e modificarne i punti, utilizzando GKey6. Un altro modo di posizionare i punti in termini di coordinate trasformazione-frame consiste, sempre dopo aver selezionato il punto da spostare, nel premere NKey: comparira' una popup che riporterà le coordinate LocX e LocY (location X e Y), ossia frame

e percentuale di trasformazione. Nel caso del nostro esempio dovremmo assicurarci che un punto della curva di Bezier abbia le coordinate (50;0.2). Ripetendo queste operazioni anche sulle altre key create, potremo creare animazioni molto complesse. Per vedere nella IPO window tutte le curve create, basta selezionare a sinistra tutte le scritte key1, key2, etc. . . tenendo premuto contemporaneamente il tasto SHIFT.

Se si desidera utilizzare la stessa curva per pi' u key e' possibile fare le operazioni di copia ed incolla. La procedura e' semplice. Poniamo il caso di aver creato cinque key, e di aver gia' impostato quattro curve di controllo. Volendo riutilizzare la prima curva di controllo anche per la key5 si eseguono le seguenti operazioni:

Si seleziona la key1

Si preme l'iconcina della freccia verso il basso, nella header della IPO window. Questo copiera' la curva nel bu#er.

Si seleziona la nuova key, la key5

Si preme l'iconcina della freccia verso l'alto, nella header della IPO window. Questo incollera' la curva nel buffer.

## 7. Radiosity

Manuel Bastioni, mbastioni@tiscalinet.it

2002-12-08 01:45:10

Contenuto della sezione

### 7.1 Cosa é la radiosity

Negli ultimi anni la computer grafica si e' mossa sempre di piu' verso la resa fotorealistica delle scene: attingendo ora dalla fisica, ora dall'arte, sono stati sviluppati degli algoritmi in grado di avvicinarsi il piu' possibile (nei limiti di tempi di computazione umani) alla realta'.

Una delle cose piu' complesse, forse la piu' complessa in assoluto, e' la simulazione del comportamento della luce. La scienza ha costruito un modello preciso di cio' che accade realmente, ma tradurre il tutto in calcoli sul computer significherebbe, nelle condizioni attuali, mesi di lavoro continuo...anche per un singolo fotogramma!

Questo significa che per avere una scena credibile occorre aspettare l'avvento dei computer quantici?

Absolutamente no!

L'occhio umano si accontenta piu' che abbondantemente di una decente approssimazione della realta'. Non occorre affatto calcolare con totale precisione il comportamento di ogni singolo fotone per raggiungere degli eccellenti risultati fotorealistici. Basta usare buoni modelli matematici che tengano conto dell'essenziale. La radiosity è uno dei migliori. Sostanzialmente i modelli piu' usati sono tre: raytracing, scanline e radiosity. Il primo, forse il piu' famoso, simula il comportamento dei raggi di luce che giungono alla telecamera: questi raggi rimbalzano e

attraversano gli oggetti trasparenti, oppure si bloccano addosso ad oggetti opachi, e quando giungono alla telecamera portano con loro tutte le informazioni accumulate.

Questa misera spiegazione e' sufficiente a far capire, almeno intuitivamente, perche' il raytracing e' adatto al rendering di superfici lisce, delle riflessioni, rifrazioni e ombre.

Ma forse suggerisce anche perche' il raytracing non e' adatto a tutto: non tiene conto di quello che accade alla luce indiretta, a tutti quei milioni di raggi che non finiscono direttamente nell'occhio dell'osservatore (la telecamera), ma che rimbalzano qua e la', portandosi dietro i colori e attenuando le ombre (che nel raytracing puro sono nettissime).

Anzi, occorre persino aggiungere che nella realta' il raytracing e' piuttosto raro: si manifesta ad esempio in una limpida giornata di sole, dove la luce diretta e' talmente forte da rendere trascurabile l'effetto delle luci interdiffuse, ma gia' il passaggio di una nuvola lo rende inadeguato alla situazione...

Al contrario il calcolo della luce interdiffusa e' il modello che si presenta piu' spesso in alcuni tipi di ambienti, in particolare negli interni delle abitazioni: le pareti sono sempre relativamente vicine, e la luce vi si riflette vicendevolmente prima di giungere alla camera; vi sono spesso fonti luminose non perfettamente direzionali, come una finestra non direttamente attraversata dai raggi solari (ma sempre luminosa)...

La radiosity e' appunto il calcolo della luce interdiffusa.

## 7.2 Qualche dettaglio tecnico

Abbiamo appena definito la radiosity come il calcolo della luce interdiffusa nell'ambiente. E' chiaramente una definizione di tipo ingenuo, ma quello che ci serve per ora e' capire il concetto a grandi linee.

Si tratta di un algoritmo sviluppato verso la fine degli anni ottanta da Donald

Greenberg, Cindy Goral ed altri presso la Cornell University e sostanzialmente e' una trasposizione in termini di energia luminosa delle teorie usate sino ad allora per il calcolo della propagazione, assorbimento ed emanazione di calore in ambienti chiusi.

Vediamo meglio

di cosa si tratta, anche se inizialmente con un linguaggio molto poco scientifico.

Immaginiamo tre pareti di tre colori differenti:

rosso verde e bianco. Da una lampada bianca parte un raggio luminoso, che ha un'energia pari a 100 (non stiamo a puntualizzare con le unita' di misura per ora). Questo raggio arriva sulla parete rossa: in parte viene assorbito, ed in parte rimbalza...non so...diciamo che rimbalza con un'energia residua di 40 (quindi 60 unita' sono state gia' assorbite... tenete presente che sto sparando numeri a caso...). Solo che ora il raggio, essendo stato riflesso da una parete rossa, ha

assunto anch'esso una colorazione rossa (cioè le radiazioni "rosse" sono state riflesse, le altre in gran parte assorbite); ed ecco che questa luce "residua" arriva sulla parete verde: ovviamente anche questa volta in parte viene assorbita ed in parte viene riflessa. Tanto per avere un'idea, diciamo che stavolta l'energia riflessa è di 15 unità. Il colore del raggio luminoso, dopo quest'ultimo "rimbalzo", avrà però cambiato di nuovo colore: le radiazioni "verdi" sono state riflesse, mentre quelle rosse rimanenti dopo l'incontro con la parete rossa sono state assorbite dalla parete verde...

Alla

fine verrà assorbita e magari solo in minima parte riflessa ancora una volta dalla parete bianca, e l'energia luminosa sarà così completamente esaurita. Ma nel frattempo la zona illuminata sulla parete verde avrà una leggera colorazione rossa, e quella illuminata sulla parete bianca una leggera illuminazione verde-rossa...

Questo processo nella realtà si ripete per milioni di volte, e dà luogo alla luce diffusa con più o meno intensità e con tutte le infinite sfumature di colori che contribuiscono al realismo.

Sono chiare le differenze con il modello del raytracing. Una delle principali, molto importante ai fini dell'uso artistico della radiosity è che l'effetto calcolato con quest'ultima è indipendente dal punto di vista. Nel caso di un filmato questo si traduce in un enorme vantaggio, perché la radiosità non deve essere ricalcolata ad ogni frame (cosa invece necessaria per il raytracing).

Questo è già un grande vantaggio. Ma c'è di più: dopo la computazione della radiosità è possibile utilizzare delle tecniche di shading molto semplici, quale ad esempio il flat shading. Questo in soldoni vuol dire che potremmo avere ambienti illuminati con la radiosity anche in applicazioni real time, ad esempio in un gioco realizzato con le OpenGL! Ve lo immaginate Tomb Raider con l'illuminazione fotorealistica basata sulla radiosità? Certo, occorre un piccolo trucco: usare il rendering scanline per mostrare i risultati della radiosità. Si tratta di un'idea validissima, che permette di renderizzare scene complesse con tutta la rapidità del metodo scanline. Ovviamente Blender lo fa alla grande...

Prima di vedere come avviene il tutto però, occorre avere un'idea approssimativa del sistema di rendering utilizzato.

## 7.3 I tre sistemi di rendering

La famigerata "graphics pipeline", tanto spesso citata parlando di rendering, è sostanzialmente un procedimento basilare di questo tipo:

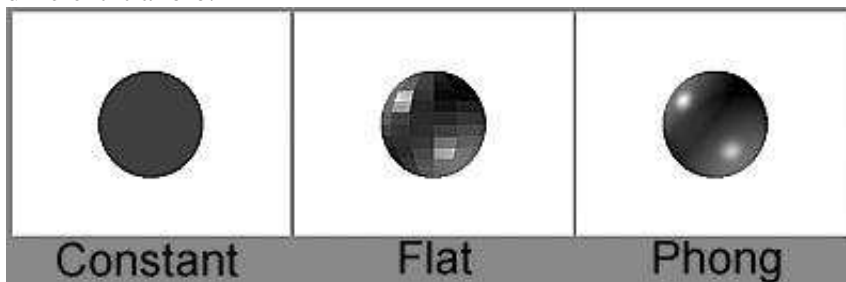
*Modeling → Viewing → Rendering*

Durante questo processo vengono affrontati alcuni problemi caratteristici, come la rimozione delle superfici nascoste,

il comportamento della luce, la proiezione di ombre. I tre differenti approcci a questi problemi hanno portato allo sviluppo di tre tecniche diverse, ognuna con vantaggi e svantaggi rispetto alle altre. Si tratta dei metodi scanline, raytracing e radiosity. Il primo, il metodo scanline, ci interessa particolarmente, perché è quello utilizzato da Blender, ed è quello che tanto ha fatto e fa discutere gli artisti blenderiani, divisi tra chi lo vorrebbe sostituito da un motore raytracing based e chi invece ne apprezza la velocità e la duttilità'.

## 7.4 Il rendering scanline

Visto che si tratta dell'output di Blender, è necessario averne una conoscenza un po' più approfondita. Lo scanline è il più antico, e per certi versi insostituibile, algoritmo usato per il rendering. È infatti ancora presente, almeno come opzione in tutti i maggiori pacchetti di grafica 3D. Spesse volte non è presentato con il nome di scanline, ma con quello di Phong, Gouraud, Flat o Constant shader. Si tratta di variazioni sul tema, con rese ed utilizzi molto differenti tra loro.



Sostanzialmente questi metodi si differenziano per il modo di trattare i quattro vettori fondamentali del rendering: vettore normale, vettore luce diffusa, vettore luce speculare e vettore luce ambiente. Data una superficie ed un modello di illuminazione, in teoria si potrebbe calcolare il colore di ogni singolo pixel dell'immagine basandosi sul calcolo del vettore uscente da ogni punto della superficie.

In teoria, beninteso, perché in pratica questo richiederebbe un carico computazionale spaventoso.

Si ricorre allora ad una prima semplificazione: le superfici vengono approssimate e divise in poligoni piatti; ognuno di questi poligoni ha un'illuminazione costante per tutta la sua estensione, ossia ha un unico vettore normale che rimane costante in tutti i suoi punti. In questo modo il numero dei vettori si riduce notevolmente, ed è possibile un calcolo relativamente rapido della scena.

Vediamo nella figura lo stesso oggetto in tre diversi rendering eseguiti tutti con Blender e quindi con il sistema scanline. Usando il constant shading l'oggetto viene renderizzato con una tinta costante. È un effetto a volte impiegato per il toon effect, ossia la resa dei cartoni animati. Nel flat shading invece le normali degli spigoli non vengono

interpolate, e i poligoni con cui e' stata approssimata la superficie rimangono ben evidenti. Nel terzo esempio, Blender utilizza l'algoritmo di Phong. E' notevole il modo in cui questo algoritmo riesca a rendere le superfici come se avessimo fatto veramente il calcolo della normale di ogni punto, quando in realta' abbiamo usato un numero di poligoni veramente basso. Il segreto sta trattare le normali dei poligoni adiacenti, realizzando una sorta di superficie media

## 7.5 Lo scanline Phong e Goraud

Gli effetti più realistici dello scanline si ottengono usando due algoritmi principali: Goraud e Phong.

Il primo, dovuto a Henry Goraud

(Henry Gouraud: Computer Display of Curved Surfaces.

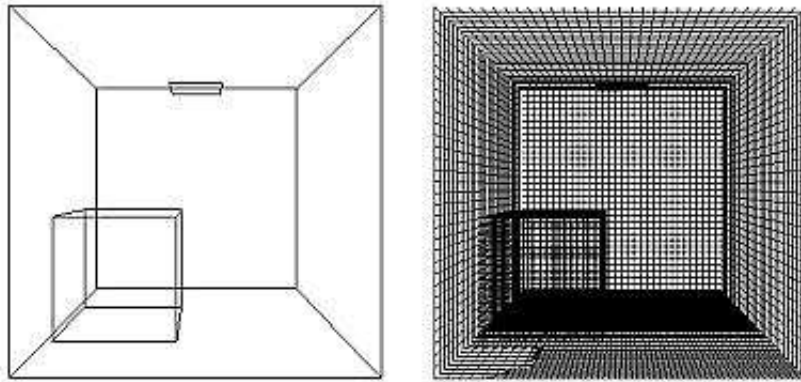
Garland Publishing, New York 1971), computa le normali medie in modo da ottenere più o meno lo stesso effetto continuo visto nel terzo esempio, ma non supporta il calcolo delle ombre. Viene usato, per la sua rapidità di computazione, nelle implementazioni della grafica in real time, tipo OpenGL.

Il secondo, sviluppato da Bui Tuong Phong (Bui Tuong Phong, Illumination for computer generated pictures, Communications of the ACM, v.18 n.6, June 1975), oltre che mediare le normali calcola anche in maniera sufficientemente realistica la proiezione dell'ombra da sorgenti puntiformi. Quello di Phong e' un modello empirico, cioè non è basato su leggi fisiche, ma sull'osservazione diretta di ciò che accade. Phong ha osservato che per le superfici molto lucide il punto di luce speculare e' piccolo e l'intensità della lucentezza decresce rapidamente, mentre per le superfici più ruvide il punto luce e' ampio e sfumato. Grazie a queste considerazioni, Phong determino' la lucentezza in base ad una formula non scientifica, ma che "ad occhio" permette di raggiungere dei risultati verosimili. E' questo l'algoritmo usato proprio da Blender. Ma come può un modello così semplice rendere un calcolo radiosity? Lo vedremo nella prossima sezione.

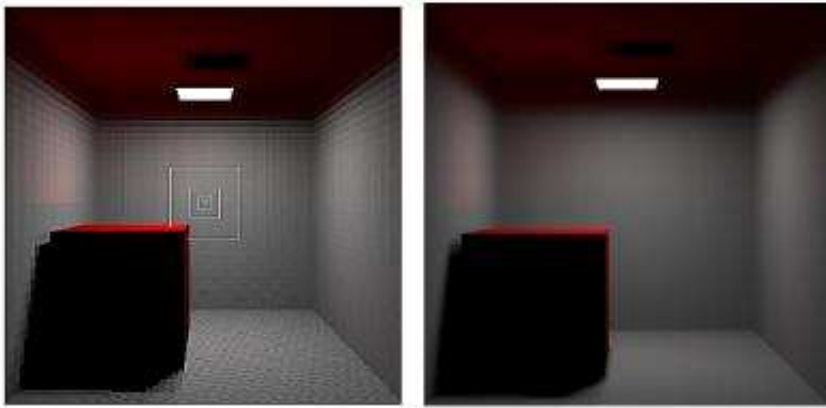
## 7.6 La discretizzazione della Radiosity

Generalmente, l'algoritmo della radiosity divide le superfici in tanti pezzettini (patches), trasformando così le superfici continue in superfici discrete, formate da tanti frammenti elementari, ognuno dei quali e' perfettamente diffusivo e di colore uniforme. Di volta in volta, ogni patch viene confrontata con le altre, per calcolarne l'energia luminosa emessa e quella ricevuta.

Laddove il fenomeno è maggiormente complesso, l'algoritmo agisce dividendo la porzione degli oggetti in pezzi ancora più piccoli.



Nella figura si vede la fitta suddivisione operata da Blender per la radiosity del semplice ambiente qui sotto.



In figura, a sinistra si vede la vista flat del risultato radiosity: ogni faccetta ha un colore ed un'emissività uniformi.

Normalmente, in altri software, questo procedimento avviene durante la fase di rendering, che a volte ha tempi interminabili.

Blender anche in questo caso si differenzia dagli altri, perché ricorre ad un sistema molto astuto che gli consente di rendere la radiosity solo con il suo semplice algoritmo di Phong.

I materiali di Blender possiedono un particolare attributo, il parametro `emit`, che regola la quantità di energia emessa da un materiale; si tratta in pratica di una sorta di fosforescenza...

Tale parametro può, ovviamente, essere settato a mano nell'editor e, al momento di fare il rendering un materiale emittente risulta visibile anche in assenza di fonti luminose vere e proprie, ossia in assenza di LAMP. Blender, dopo il calcolo della radiosity, non fa altro che assegnare ad ogni vertice un valore di emissione energia ed un colore preciso, entrambi calcolati con l'algoritmo ricorsivo radiosity. Per questo motivo poi il rendering è rapidissimo, e soprattutto possibile: si tratta del semplice rendering di vertici colorati, una cosa di routine per il modello Phong, ma usata in maniera davvero furbissima.

È quello che, tecnicamente, viene chiamato *environment preprocessing*.

Occorre notare però che tale modello è perfettamente valido solo se usato in assenza di Lamp esterne.

In caso contrario i calcoli delle riflessioni di Phong si andranno a sovrapporre agli effetti della radiosity, abbassando in tal modo la fedeltà del risultato alle leggi fisiche. Tuttavia, non è detto che questo sia un difetto: l'artista può in tal modo aggiungere effetti luminosi con il massimo controllo della scena, cosa che spesso è desiderabile

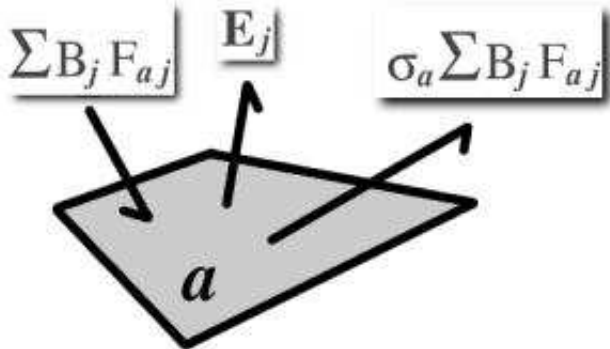
## 7.7 L'equazione della radiosity

Inizialmente non volevo addentrarmi negli aspetti matematici della cosa, ma poi mi sono reso conto che una conoscenza di questo genere è indispensabile per un uso completo dei parametri presenti nell'interfaccia di Blender. Infatti alcuni pulsanti, come "Max Iterations" o "Hemires" necessitano di una conoscenza, anche minima, dell'equazione della radiosity e delle tecniche usate per implementarla.

Questa è l'equazione della radiosity per superfici finite. È già dovuta alla serie di ipotesi sulla discretizzazione di cui abbiamo già parlato, che hanno permesso di passare da complesse equazioni differenziali sviluppate per superfici infinitesime a questa forma molto più computabile:

$$B_a = E_a + \sigma_a \sum B_j F_{aj}$$

L'equazione descrive la quantità di energia che viene emessa da una superficie, considerandola come somma dell'energia emessa dalla superficie stessa e dell'energia ricevuta e riflessa da altri oggetti circostanti. Il modello fisico è basato sull'ipotesi di un ambiente chiuso, senza scambi di energia con l'esterno, superfici opache che riflettono la luce in tutte le direzioni, secondo la legge lambertiana di diffusione, e di un mezzo (l'aria) perfettamente trasparente



- $B_a$ : Radiosità della superficie, espressa come energia per unità di area
- $E_a$ : Energia emessa dalla superficie stessa, o emissività, anch'essa energia per unità di area.
- $\sigma_a$ : Coefficiente di riflettività, compreso tra zero e uno. Indica la quantità di luce riflessa dalla superficie.
- $B_j$ : Radiosità della superficie  $J$
- $F_{aj}$ : Fattore di forma, un coefficiente compreso tra zero e uno, che tiene conto della presenza e della geometria della superficie  $J$ .

Il fattore di forma è un numero puro, e descrive il comportamento dell'energia che viene scambiata

tra la superficie computata e una superficie ad essa vicina. I fattori fondamentali in questo calcolo sono a distanza tra le superfici ed il loro orientamento nello spazio.

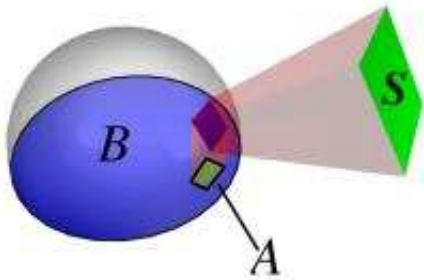
Quest'ultimo si ottiene considerando l'inclinazione del vettore normale uscente dal centro delle superfici. È sempre dal centro delle superfici vengono prese le distanze. Dall'esame degli elementi che vengono dunque considerati nel calcolo di questo fattore, risulta chiaro che il risultato è totalmente indipendente dal punto di vista.

In teoria, il fattore di forma andrebbe calcolato per superfici infinitesime; chiamando tali superfici come  $dS_1$  e  $dS_2$ , avremo per ogni coppia un fattore di forma dato da INSERIRE FORMULA [IMAGE]

Questo, nel considerare superfici di dimensioni finite, porterebbe, al solito, ad una serie di integrali dal costo computazionale elevatissimo, che renderebbe arduo il calcolo anche solo tra

due semplici piani. Quanto detto sinora dunque ha importanza solo nella preparazione teorica, poiché nella pratica si ricorre all'analogia Nusselt.

### Il modello Nusselt



## 7.8 Materiali emittenti

## 7.9 Uso dei pulsanti

Sostanzialmente, i pulsanti che bisogna conoscere per poter usare la radiosity in Blender sono questi:

### Pulsanti Base

- **Collect Meshes** Serve per inserire gli oggetti selezionati tra quelli da considerare al momento del calcolo della radiosity. Ovviamente, questo permette di eseguire la computazione solo su una parte della scena, evitando di lavorare su oggetti irrilevanti, come ad esempio quelli troppo piccoli.
- **Go** Serve per avviare il calcolo della radiosity.
- **Replace Meshes** Serve per sostituire, nella scena, gli oggetti creati dalla radiosity, suddivisi in patch emittenti, a quelli originali.
- **Add New Meshes** Serve per aggiungere, nella scena, gli oggetti creati dalla radiosity, suddivisi in patch emittenti, senza cancellare quelli originali.
- **Free Radio Data** Serve uscire dalla modalità radiosity. Si perdono in tal modo tutti i risultati dei calcoli fatti, a meno di non aver già sostituito o aggiunto le nuove mesh alla scena.

### Pulsanti di visualizzazione

E' possibile scegliere tre differenti modalità per visualizzare i risultati in real time durante durante il calcolo della radiosity. Si tratta di

- **Wire** I risultati del calcolo vengono mostrati sotto forma di reticolo, o wireframe.
- **Solid** I risultati del calcolo vengono mostrati con rendering scanline real time, di tipo flat.
- **Goraud** I risultati del calcolo vengono mostrati con rendering scanline real time, di tipo Goraud.

E' chiaro che il controllo migliore si ottiene usando il goraud, ma questo comunque non ha alcun effetto sul calcolo della radiosity.

## 7.10 Uso delle texture

## 7.11 Vantaggi e svantaggi

La radiosity ha molteplici vantaggi dal punto di vista della resa, ma anche alcuni svantaggi.

### VANTAGGI

- Qualità fotorealistica dell'immagine
- Accurata simulazione del trasferimento di energia
- Ombreggiatura realistica soft e luce interdiffusa

### SVANTAGGI

- Enorme calcolo
- Enorme infittimento della mesh iniziale
- Mancanza di riflessione tipo specchio e di rifrazione

Citando l'opinione riportata su [www.siggraph.org](http://www.siggraph.org)

*The largest single advantage of the radiosity method for computer image generation is the highly realistic quality of the resulting images. No other method accurately calculates the diffuse interreflection of light energy in an environment. Soft shadows and color bleeding are natural by-products of this method, just as hard shadows and mirror-like reflections are natural by-products of a typical ray-tracing algorithm.*

E' impressionante che Blender possieda di default uno strumento tanto potente, il cui unico reale svantaggio e' il lungo tempo computazionale richiesto. Eppure anche questo non e' un reale ostacolo in Blender, che grazie allo stratagemma del calcolo pre-rendering deve calcolare la radiosity solo una volta.

L'unico svantaggio a mio avviso e' rappresentato dall'impossibilita' di realizzare filmati con radiosity perfette quando nella scena ci sono oggetti di massa considerevole che si spostano, o peggio ancora, quando a postarsi sono le fonti emittenti. In tal caso la radiosity andrebbe ricalcolata fotogramma per fotogramma, appunto perche' cambiano posizione le sorgenti di luce interdiffusa. Questo, sebbene possibile, richiederebbe un pre-calcolo ogni volta, che in termini di tempo e di pazienza, non e' sopportabile.

Tuttavia, se a muoversi non sono le fonti radiative o oggetti con grosse superfici riflettenti, il problema e' trascurabile.

## 7.12 Webografia

Note. Attenzione, il link sugli appunti presenta un bug, per cui i collegamenti vengono alterati. Occorre fare il copia ed incolla.

- <http://www-2.cs.cmu.edu/afs/cs/user/ajw/www/software/index.html#Radiator>  
Un programma per l'apprendimento interattivo della radiosity.
- <http://www.rz.rwth-aachen.de/vr/teaching/seminars/ws00/Papers/WS00SchneiderTalk.pdf>  
Un chiaro testo molto ben illustrato.
- <http://www.scs.leeds.ac.uk/cuddles/rover/radpap.htm>  
Un incredibile sito con link di alta qualita'.

---

## 8. Python script

2002-09-26 23:15:03

Contenuto della sezione

## 8.1 Introduzione

Il python e' un linguaggio di programmazione interpretato. Questo significa che non puo' produrre applicazioni eseguibili indipendenti, bensì ha sempre bisogno di una sorta di programma 'padre' che legga il file.py ed esegua le istruzioni in esso contenute. Se i limiti sono rappresentati da questa necessita' di un software interprete, i vantaggi che hanno portato alla sua scelta anche dalla NaN sono molteplici:

- E' totalmente gratuito,
- gira su qualsiasi piattaforma: windows, linux, etc. . .
- viene usato da moltissimi utenti
- e' estensibile
- e' semplice e veloce.

La sua nascita risale al dicembre del 1989, ed e' dovuta al ricercatore Guido Van Rossum. Il suo nome non deriva da quello del fortissimo serpente, bensì da una nota trasmissione comica, che pare piacesse molto a Van Rossum: The Monty Python's Flying Circus. Nella sintassi somiglia molto al C, ma il suo utilizzo e' alquanto piu' semplice.

## 8.2 L'interprete Blender

Scaricando ed installando tutto il pacchetto Python, scaricabile da internet all'indirizzo [www.python.org](http://www.python.org), il sistema viene fornito anche di un interprete, in grado di leggere e far agire i file.py. Tuttavia nel caso di Blender non occorre utilizzare un interprete esterno: fa tutto lui. Insieme a Blender viene infatti distribuita una libreria dinamica che si chiama all'incirca *pythonx.x.dll*. Questa permette a Blender di funzionare da interprete e rende facoltativo il download del pacchetto ufficiale. L'unica cosa e' che a volte per far funzionare alcuni script molto complessi occorre utilizzare moduli avanzati che si trovano solo nell'installazione full del python originale. E' importante, dopo l'installazione, andare a configurare le path di sistema nel file *autoexec.bat* (Per quanto riguarda win 95 e 98, mentre per NT, 2000 e XP occorre seguire una procedura meno diretta), aggiungendo una linea di istruzione, ad esempio supponendo che il python si trovi in C:\Python21

```
SET PYTHONPATH=C:\Python21\Lib
```

Un metodo piu' semplice ed immediato, consiste nel settare il percorso direttamente nella infowin di Blender, nel pulsante 'Python:'.

## 8.3 La text-window

Blender e' veramente un pacchetto completo: tra i suoi svariati tipi di finestre se ne trova una per editare ed eseguire i python script. In realta' gli script possono essere realizzati con qualsiasi text-editor, ad esempio con il semplice Notepad di Windows, tuttavia l'interfaccia di Blender e' studiata appositamente per questo scopo, e facilita grandemente il compito. La console dos in questo ambito assume il ruolo di *sdout* e di finestra di debug.

Per aprire una textwindows occorre premere SHIFT-F11 oppure trovare sull'header della windows scelta l'iconcina che rappresenta un foglio bianco con delle righe. A questo punto saremo pronti per la scrittura del nostro codice python. Osserviamo che in basso troviamo un pulsante con la scritta *Screen 12*, che indica la grandezza del font visualizzato. Volendo usare caratteri

piu' grandi, e' sufficiente premere su questo bottone e scegliere Screen 15. Il pulsante subito a sinistra di questo serve invece per fare due operazioni ben piu' importanti: creare un nuovo script o aprirne uno da un file esterno (ADD NEW o OPEN NEW). In entrambi i casi, ad operazione avvenuta, compare anche una crocetta, che serve a cancellare lo script dalla memoria di Blender (ma non si cancella il file).

## 8.4 Operazioni in text-window

Con i seguenti shortcut si eseguono le operazioni di editazione:

- ALT-C Copia il testo selezionato nel buffer,
- ALT-X taglia il testo selezionato nel buffer,
- ALT-V incolla il testo selezionato dal buffer,
- ALT-O apre un file di testo,
- SHIFT-ALT-F apre un men'ù con le operazioni sui file,
- ALT-J appare una finestra per andare ad un determinato numero di riga,
- ALT-P esegue lo script,
- ALT-U infiniti undo,
- ALT-R ripristina undo,
- ALT-A seleziona tutto il testo.
- CTRL-ALT-F appare la popup per cercare determinate parole.
- ALT-F continua la ricerca di una determinata parola.

## 8.5 L'importanza dei rientri

Mentre altri linguaggi, ad esempio il C++, o il linguaggio di POV-ray usano le parentesi per separare le varie istruzioni, il python usa i rientri di margine (indentazione): tutte le istruzioni con la medesima distanza dal margine sinistro sono considerate comprese in un'unica istruzione. Questo particolare e' molto importante, perche' a volte messaggi di errore in righe apparentemente corrette sono dovuti a spazi errati o al mischiare tabulazioni e spazi semplici.

## 8.6 Le basi

La prima riga che deve sempre essere contenuta e'

```
import Blender
```

che ordina a Blender di importare il modulo fondamentale contenente le sue API. Dopo questa riga possiamo gia' usare gli usuali comandi del Python e vedere il risultato nella finestra console premendo ALT-P. Ad esempio le righe

```
import Blender  
print 2+2
```

visualizzano il risultato dell'addizione nella finestra di console. Una qualsiasi riga preceduta da "#"

viene ritenuta un commento. La text-windows non supporta i caratteri speciali della tastiera italiana, per cui occorre ricorrere alla sequenza ALT-35 per ottenere il carattere "#", ALT-9-1 per il carattere '[' e ALT-9-3 per ']'.

## 8.7 I Moduli

Il python fa largo uso di moduli, anzi potremmo dire che questi rappresentano la sua filosofia d'uso. Essi non sono altro che altri file esterni, sempre .py, che contengono funzioni particolari o molto complesse. E' semplice creare dei moduli; ad esempio se all'interno di uno script imposto una funzione in questo modo:

```
def area_tr (base, altezza):  
    a = (base * altezza)/2  
    print a
```

e poi salvo il file in formato .py, chiamandolo, ad esempio, Formule.py, posso richiamarlo in uno script successivo con il comando import, utilizzandolo come modulo:

```
import Blender  
import Formule
```

e posso utilizzare tutte le funzioni in esso definite, come in questo caso:

```
import Blender  
Formule.area_tr(5, 2)
```

La finestra console mi mostrerà il valore dell'area di un triangolo di base 5 e altezza 2

## 8.8 I moduli Builtin

Alcuni moduli sono particolarmente importanti, e contengono le funzioni basilari che devono essere svolte dal Python. Si tratta dei moduli costruiti internamente, o builtin, che sono dunque sempre presenti, anche se non vengono chiamati con l'istruzione import.

## 8.9 Il modulo sys

Il modulo sys e' uno di questi, e contiene diverse informazioni sulle variabili di sistema.

```
['.\\DLLs', '.\\lib', '.\\lib\\plat-win',  
'\\.\\lib\\lib-tk', 'C:\\Programmi\\B14', 'C:\\  
Programmi\\B14', '']
```

## 8.10 Funzioni matematiche

Nel banale esempio precedente abbiamo utilizzato una funzione matematica elementare: l'addizione, unita al comando print che ha stampato il risultato nella console di Blender. Gli operatori matematici standard sono + - \* /, per cui eseguendo, ad esempio

```
import Blender  
print 3*2
```

si ottiene come risultato il numero 6. Il python tratta senza problemi anche i numeri immaginari, per cui si pu' tranquillamente eseguire

```
import Blender  
print (1 + 2j)*3j
```

ottenendo come risultato il numero complesso (-6 + 3j)

## 8.11 Funzioni generali

### PRINT

La funzione print e' utilizzata per stampare nella console di output un determinato valore: ad esempio

```
print "pippo"
```

restituisce pippo in output. E' possibile stampare piu' argomenti su di una sola riga, utilizzando delle virgole, ad esempio:

```
print "pippo", "pluto"
```

restituisce pippo pluto in output. Oltre alle stringhe vengono stampati anche i valori delle variabili di qualsiasi tipo, per cui il codice

```
a=10
```

```
print "pippo", "pluto",a
```

Restituisce pippo pluto 10 in output.

## 8.12 Strutture di dati

Le principali strutture di dati, in python, sono tre:

Liste

Tuples

Dizionari

La lista e' un vettore riga formato da piu' elementi indicati tra le parentesi e separate da una virgola. Vediamo un esempio:

```
lista=[8,2,4,6]
```

Con questo semplice comando, abbiamo formato una lista contenente quattro valori numerici. Possiamo accedervi sapendo che implicitamente il python assegna ad ognuno degli elementi un indice, per cui il primo elemento e' associato al numero 0, il secondo al numero 1 e cosi' via. . . Ad esempio, il comando

```
print lista[0]
```

ordinera' all'interprete di stampare il primo elemento, per cui vedremo apparire in output il numero '8'.

Una lista python pu' contenere tranquillamente sia stringhe che numeri, con l'accortezza che le prime vanno indicate tra apostrofi per non confonderle con delle variabili. Ad esempio:

```
lista=['pippo','pluto',3,5,'paperina']
```

## 8.13 Accesso alle liste

Vi sono alcuni comandi e metodi che permettono di manipolare con facilità le liste, ad esempio il comando len( nome della lista) permette di contare quanti elementi essa possiede al suo interno. Ad esempio

```
lista=['pippo','pluto',3,5,'paperina']
```

```
print len(lista)
```

fornisce come risultato 5. Vi sono poi molti altri metodi; ad esempio facendo riferimento ad una lista di nome mialista:

```
mialista[i]=b
```

Sostituisce l'elemento esistente nella posizione i con il nuovo valore b

```
mialista[i:u]=b
```

Taglia i valori di mia lista a partire dall'indice i sino

all'indice u e li sostituisce con l'unico valore b  
del(i)  
mialista.insert(a,i) Inserisce l'elemento a nella posizione i della lista.  
mialista.append(a) Inserisce l'elemento a in coda alla lista  
mialista.index(a) Restituisce il numero di posizione occupato dall'elemento a della lista  
mialista.remove(a) Rimuove dalla lista il primo elemento a trovato  
mialista.sort() Ordina la lista in maniera crescente  
mialista.reverse() Ordina la lista in maniera decrescente  
mialista.count(a) Restituisce il numero di volte che l'elemento a e' presente nella lista.

## 8.14 I cicli in python

## 8.15 Gli oggetti di Blender

In Blender tutto e' organizzato tramite gli oggetti. l'oggetto e' dunque alla base della gerarchia delle scene. Ogni elemento, sia esso una luce, una mesh, o altro, e' un oggetto, e pertanto possiede delle proprieta' standard. Per accedere agli oggetti occorre utilizzare il modulo 'Object'. Per quanto segue faremo riferimento al modulo python ver. 1.8, che attualmente e' ancora lo standard 4 . Per convenzione indicheremo tra parentesi quadrate gli argomenti opzionali, mentre le parentesi tonde appartengono alla sintassi.

## 8.16 Le funzioni standard di Object

Get([name])

Se 'name' viene specificato, la funzione restituisce l'oggetto richiesto, altrimenti creera' una lista con tutti gli oggetti presenti nella scena. Se si cerca uno specifico oggetto che non viene pero' trovato, la funzione restituisce il valore 'none' .

GetSelected()

Crea una lista con tutti gli oggetti selezionati in quel momento. l'oggetto attivo (cfr. pag. 21) e' il primo della lista.

Update(name)

Aggiorna l'oggetto che si chiama 'name'. Si tratta di una funzione sperimentale

verts, che restituisce una lista di vertici per la faccia considerata

mats, che restituisce una lista dei nomi dei materiali

has\_col, ag che indica se i vertici della mesh sono colorati

has\_uvco, ag che indica se la mesh ha le coordinate UV

creata per combattere il ritardo nell'aggiornamento della 3D view.

Un paio di esempi

Il seguente esempio restituisce la lista ed il numero di oggetti presenti in una scena:

```
#version windows
```

```
import Blender
```

```
import sys
```

```
obj=Blender.Object.Get()
```

```
print
```

```

print obj
print len(obj)
print
sys.stdout.flush()
che nella versione linux diventa
#version Linux
import Blender
obj=Blender.Object.Get()
print
print obj
print len(obj)
print

```

Una volta 'preso' un oggetto con il metodo Get(name) possiamo accedere a molte delle sue caratteristiche grazie agli oggetti name, che restituisce il nome dell'oggetto considerato. LocX, che restituisce la coordinata x dell'oggetto. LocY, che restituisce la coordinata y dell'oggetto. LocZ, che restituisce la coordinata z dell'oggetto.

## 8.17 il modulo NMesh

Da descrivere. . .

```
GetRaw([name])
```

Il metodo GetRaw([name]) di NMesh permette di accedere o di creare una nuova mesh. Se si specifica il nome dell'oggetto esistente, questo viene preso ed utilizzato, altrimenti viene creato un nuovo oggetto privo di elementi. Ad esempio, il codice

```
import Blender
```

```
from Blender import NMesh
```

```
my_raw=NMesh.GetRaw("pippo")
```

memorizza nella variabile my\_raw tutte le caratteristiche dell'oggetto il cui meshname e' 'pippo'. E' poi possibile accedere agli elementi base, quali facce e vertici, mediante gli oggetti

name, che restituisce il nome della mesh considerata.

faces, che restituisce una lista contenente tutte le facce della mesh considerata.

verts, che restituisce una lista di vertici per la faccia considerata

mats, che restituisce una lista dei nomi dei materiali

has\_col, ag che indica se i vertici della mesh sono colorati

has\_uvco, ag che indica se la mesh ha le coordinate UV

ad esempio, il codice seguente, stampa nella console il numero di facce posseduto dall'oggetto il cui meshname e' 'pippo'

```
import Blender
```

```
from Blender import NMesh
```

```
skin=NMesh.GetRaw("pippo")
```

```
skin_faces=skin.faces
```

```
print len(skin_faces)
```

Esempio: La creazione di una mesh

Come sappiamo una mesh (letteralmente, rete) e' formata da una lista di facce, ed ogni faccia, a sua volta, e' formata da una lista di vertici. Facce e

vertici sono elementi di pertinenza al modulo NMesh. Precisamente possiamo definire un vertice v tramite le sue coordinate con il comando

```
v=NMesh.Vert(x,y,z)
```

Allo stesso modo possiamo creare una faccia f (senza vertici), con il comando

```
f=NMesh.Face()
```

Facciamo un esempio piu' concreto. Iniziamo dal canonico

```
import Blender
```

poi da Blender inportiamo il modulo NMesh

```
from Blender import NMesh
```

Questo modulo fondamentale ci servira' sia per creare una nuova mesh (priva di vertici) di nome new\_mesh con il comando

```
new_mesh = NMesh.GetRaw()
```

Gia' che ci siamo, specifichiamo anche che la nuova mesh non ha vertici colorati ne' mappatura UV:

```
# flag che indica che non ci sono coordinate uvmapping
```

```
me.has_uvco=0
```

```
# flag che indica che non ci sono facce colorate
```

```
me.has_col=0
```

Sappiamo che ogni mesh possiede al suo interno una lista di vertici (in questo caso ancora vuota) chiamata 'Vert'; l'operazione base e' quella di definire i vari vertici ed utilizzare il comando append visto in precedenza per aggiungerli di volta in volta alla lista 'Vert'. In questo caso faremo un quadrato, utilizzando sempre la stessa variabile v, che tanto viene ridefinita ogni volta:

```
v=NMesh.Vert(1.0,0.0,0.0)
```

```
new_mesh.verts.append(v)
```

```
v=NMesh.Vert(1.0,1.0,0.0)
```

```
new_mesh.verts.append(v)
```

```
v=NMesh.Vert(0.0,1.0,0.0)
```

```
new_mesh.verts.append(v)
```

```
v=NMesh.Vert(0.0,0.0,0.0)
```

```
new_mesh.verts.append(v)
```

Ora tutti i vertici sono memorizzati nella lista verts della nuova mesh,

tuttavia non esistono ancora facce che li utilizzino, e Blender non sa in quale ordine prenderli per costruire il nostro oggetto. Creiamo quindi un nuovo oggetto di tipo faccia, oggetto che appartiene anch'esso al modulo NMesh:

```
f=NMesh.Face()
```

Ora diamo a questa faccia, che abbiamo memorizzato nella variabile f, i vertici creati poco fa:

```
f.v.append(new_mesh.verts[0])
```

```
f.v.append(new_mesh.verts[1])
```

```
f.v.append(new_mesh.verts[2])
```

```
f.v.append(new_mesh.verts[3])
```

Ora non ci rimane altro da fare che mettere la nuova mesh nella scena,

utilizzando il comando PutRaw(mesh, [name, renormalise]) nel modo seguente:

```
NMesh.PutRaw(new_mesh, " Piano", 1)
```

Finalmente, ordiniamo a Blender di ridisegnare tutta la scena, in modo da aggiornarla:

```
Blender.Redraw()
```

Costruzione di una Mesh Complessa

Passiamo ora a qualcosa di piu' difficile e stimolante, la creazione di una mesh

ben piu' complessa. l'algoritmo che stiamo per introdurre e' alla base degli script di JM Soler, interessantissimi dal punto di vista didattico. Al momento in cui scrivo il sito di JM si trova in . . .

Ricordiamo brevemente che i cicli in python sono otti-mizzati

per le liste. Essi si eseguono nel seguente modo:

```
for questa_variabile in questa_lista :
```

```
comando_1
```

```
comando_2
```

```
...
```

```
comando_n
```

Attenzione al rientro dopo for e ai due punti!

Il problema fondamentale della creazione di un oggetto mesh risiede nel fatto che ogni faccia ha dei vertici in comune con tutte le altre facce ad essa adiacenti. Immaginiamo ad esempio una mesh quadrata formata da quattro facce. In tutto, per descriverla completamente, sono sufficienti 9 vertici, che potremmo memorizzare in una lista cosi' indicizzata

```
[0,1,2,3,4,5,6,7,8]
```

tuttavia, ogni faccia contiene quattro vertici, per cui la nostra mesh, come somma di quattro facce, dovrebbe contenere 16 vertici. Evidentemente 7 sono ridondanti. Su mesh complesse questa ritondanza pu' generare migliaia di vertici inutili.

---

## 9. Game Engine

2002-11-01 11:57:38

Contenuto della sezione

### 9.1 Introduzione

-Il GameEngine è un ottimo strumento ideato per la creazione di materiale interattivo (videogiochi,dimostrazioni varie etc.). Associando, l'agevolezza della modellazione in Blender Creator , la possibilità di animare gli oggetti del gioco (GameObjects) tramite le IPO curves,la possibilità di animare le loro mesh tramite le armature,l'applicazione diretta delle texture,l'aggiunta delle luci e dei suoni con la logica e la dinamica del GameEngine si ottiene un ottimo prodotto per la creazione di qualsiasi contenuto interattivo.Tutto questo è reso ancora più flessibile dalla possibilità di espanderlo, programmando il comportamento degli oggetti con un linguaggio semplice, e per di più free, qual è il Python.

-La logica del funzionamento del GameEngine è molto semplice.Qualsiasi oggetto che sta in un world ha delle proprietà fisiche e dinamiche proprie nei confronti di quest'ultimo.Il quale può essere gestito dai LogicBricks(mattoni logici) che funzionano secondo questa logica:

**Evento** connesso a- **Oggetto** connesso a - **esecutori** (Sensors----Controllers---- Actuators)

-Gli eventi(Sensors)sono l'input della tastiera,i tasti del mouse o il suo movimento,collisioni, etc. Gli oggetti (Controllers) sono tutti coloro che controllano l'evento.Alla fine gli esecutori fanno si che succeda qualcosa (spostarsi in qualche direzione,ruotarsi,far partire un animazione un suono etc.)quando un certo evento accade.Adesso entriamo in dettaglio di quello che abbiamo detto fin ora.

## 9.2 Le opzioni del Game engine

-Premendo su Game sopra il menu della finestra Info si apriranno queste scelte:

### **Start Game**

-Inizia il GameEngine(si ferma premendo Esc)

### **Enable All Frames**

-Con questa scelta selezionata il GameEngine funziona senza perdere nessun frame.Utile quando si registrano animazioni in sequenza d'immagini,o quando si ha l'impressione che nel nostro lento computer si perde qualche cosa nel calcolo nelle collisioni.

### **Show framerate and profile**

-Durante il funzionamento, il GameEngine rende visibile a quanti frame al sec. sta lavorando e dei dati sulla distribuzione del lavoro.

### **Show debug properties**

-Le proprietà degli oggetti selezionate possono essere visualizzate in tempo reale per essere controllate.

### **Autostart**

Attiva lo start automatico per la scena.

---

## 9.3 Le opzioni della finestra info

-Sulla finestra Info (contiene un i sull'icona)si possono attuare delle scelte permanenti.Dopo aver scelto si possono salvare con Ctrl + U

### **Vertex Arrays**

-Disattiva le vertex arrays.Accelemano la velocità del calcolo delle scene complesse. Se il vostro OpenGL non lo sopporta va disattivato.

### **No sound**

-Disattiva l'emissione dei suoni.

### **No mipmaps**

-Disattiva il mipmap delle texture.Accelema il gameengine ma diminuisce la qualità della renderizzazione delle texture.

### **Texture**

-Si può inserire un path ove Blender cercherà file immagine.

### **Python**

-Si può inserire un path ove Blender cercherà moduli addizionali.

### **Sound**

-Si può inserire un path ove Blender cercherà file audio.

---

## 9.4 Bottoni nella finestra real-time

-In Blender come in realtà il comportamento degli oggetti dipende dalle forze applicate,dall'attrito tra le superfici o con l'ambiente circostante,dalla gravità,dalla massa di ogni oggetto o dal materiale con cui è composto.Vediamo più in dettaglio i vari modi di settare la dinamicità di un oggetto.

-Quando Blender è aperto premendo F8 nella parte sotto la finestra 3D (se l'apertura è quella di default)si aprirà la finestra del real-time.Questa sarà una delle finestre che tratteremo maggiormente in questo documento. .Questo è il cuore del GameEngine .Per questo motivo è bene prestare attenzione a quello che contiene a cosa servono e come s'impostano.

-Nella parte sinistra della finestra in alto c'è il bottone Actor.Premendo sopra, altri bottoni

saranno visibili. Vediamoli:

### **Actor**

-Selezionando questo bottone l'oggetto è valutato dal gameengine.

### **Ghost**

-Fa sì che l'oggetto non restituisca collisione. Quando un oggetto e Ghost si può trapassare senza difficoltà, però riesce a registrare la collisione. Può servire nel caso si voglia avviare un Sensore senza collidere.

### **Dynamic**

-Selezionando Dynamic l'oggetto è sottoposto a tutte le leggi della fisica che abbiamo menzionato sopra. Inoltre altri bottoni diventano visibili.

### **Rigid Body**

-Per capire il funzionamento di quest'opzione è meglio fare un esempio. Abbiamo una sfera alla quale applichiamo una forza orizzontale. In realtà alla presenza d'attrito la sfera avanzando, ruoterà intorno a se stesso. In Blender alla presenza di attrito la sfera si sposterà solamente in orizzontale senza ruotare. Selezionando questo bottone anche in Blender la sfera ruoterà intorno a se stesso.

### **DoFh**

-Se selezionato, l'oggetto si muove avendo i parametri impostati nella finestra dei materiali (vedere 3.1.2)

### **RotFh**

-Ruota l'oggetto secondo la superficie sopra la quale si muove. Agisce insieme ai parametri nella finestra dei materiali

(per fare sì che un oggetto segua la superficie sulla quale si muove, selezioniamo quest'ultima, andiamo nella finestra dei materiali (capitolo 3.1.2). Impostiamo Fh Dist a un valore 0.10 (più facce ha la superficie più piccolo è questo valore ed in più dovrebbe avere un size uguale a 1). Poi selezioniamo l'oggetto e clicchiamo sul RotFh. Anche l'oggetto deve avere il size 1)

### **Mass**

-la massa dell'oggetto.

### **Size**

-la grandezza della sfera che avvolge l'oggetto. Il perimetro esterno della sfera determina l'area della collisione dell'oggetto. Se dentro quest'area si trova un altro oggetto c'è stata una collisione.

### **Form**

-Aiuta a controllare il comportamento del Rigid Body

### **Damp**

-Determina l'attrito con l'ambiente circostante. Notate che sotto la forza della gravità un oggetto non cade più velocemente se la sua massa è più grande ma se attrito con l'ambiente che lo circonda è più piccolo. Con questo si possono simulare ambienti come lo spazio, l'aria o l'acqua solamente cambiando il valore.

### **RotDamp**

-Stesso come damp però è valido per la rotazione.

### **Anisotropic**

-Questo bottone, se selezionato rende visibili tre altri bottoni che rappresentano l'attrito fra gli oggetti uno con l'altro. Si può inserire un valore per ogni asse. Molto utile nei casi in quali gli oggetti hanno un movimento non uguale nelle tre direzioni. Per es. uno skateboard cammina abbastanza facilmente avanti o indietro ma lateralmente non è la stessa cosa.

---

## 9.5 Bottoni nella finestra dei materiali

-Ora facciamo un salto nella finestra dei materiali.In questo modo completiamo l'impostazione dinamica dell'oggetto. Premendo F5 nella sotto la finestra 3D si apre la finestra dei materiali.Dobbiamo aggiungere un materiale al nostro oggetto,premendo sull'icona con il segno - (meno).Nella finestra di pop-up che si apre confermiamo Add New.Una serie di bottoni appare. Quelli che interessano a noi sono nella parte sinistra in alto. Clickando sopra il bottone DYN una parte dei bottoni cambia.Sono quelli che ci interessano per il nostro obiettivo.Li elenchiamo qui sotto spiegando come si usano e a cosa servono.

### **Fh Norm**

-Selezionando questa opzione l'oggetto riceverà una forza nella direzione della normale nelle discese o salite.Avendo un oggetto in cima ad una salita se Fh Norm e selezionato, scivolerà anche se non agisce nessun'altra forza su di esso.

### **Fh Dist**

-Questa è l'area nella quale la Fh agisce.

### **Fh Damp**

-Controlla l'attrito dentro l'area Fh dist

### **Restituite**

-Questo slider determina l'elasticità dell'oggetto.Un valore pari a uno significa che l'oggetto trasforma tutta l'energia cinematica in una forza con direzione opposta con la direzione del suo movimento.Questo è un corpo con elasticità ideale.Si può paragonare ad una palla che continua a rimbalzare per sempre.

### **Friction**

-L'attrito causato dal materiale dell'oggetto.Un valore basso potrebbe simulare la scivolata su ghiaccio.

### **Fh Force**

-Questa forza fa fluttuare l'oggetto sopra una superficie.

## 9.6 Proprieta' dell'oggetto

- Torniamo ancora una volta nella finestra real-time(F8).Nella parte sinistra in basso c'è un bottone grande lungo denominato Add. Ogni click sopra di esso aggiunge una proprietà all'oggetto.Le proprietà agiscono come variabili locali.Possono essere modificate in corsa tramite i Sensor Property ed Actuator Property.Appena premuto su Add sotto di esso appaiono cinque campi che definiscono la proprietà.

- Il primo è **Del**.Un bottone con il quale potete cancellare questa proprietà

- Il secondo è il tipo della proprietà. I tipi possibili sono:

### **Bool**

- Crea una proprietà di tipo boolean. I valori sono TRUE o FALSE.Quando viene usato nei Sensors ,Actuators e nelle espressioni nei Controollers assicuratevi di scrivere in lettere capitali.

### **Int**

-Crea una proprietà di tipo Intero.Vale a dire dei numeri interi che vanno da 2147483647 a 2147483647

### **Float**

- Crea una proprietà di tipo Float.Vale a dire numeri che contengono la parte decimale dopo la virgola.

### **String**

-Crea una proprietà di tipo String.In pratica del testo

### **Timer**

-Inizia un Timer che parte dal momento che l'oggetto è stato creato.Si misura in secondi

- Il terzo campo è il nome della proprietà.Le maiuscole non sono uguali con le minuscole.
  - Il quarto campo è il valore della proprietà
  - Il quinto è un pulsante di debug.Se selezionato, il valore della proprietà è visibile anche durante il funzionamento del GameEngine.Molto utile nel caso si voglia verificare il suo valore durante lo svolgimento del gioco.
  - Per modificare i valori delle proprietà, premere sul campo desiderato tenendo premuto Shift e cambiare il valore con la tastiera.Si può anche premere con il mouse e tenendo premuto muovere il mouse.In quest'ultimo modo però non si ottengono risultati molto precisi.
- 

## 9.7 Introduzione ai LogickBricks e Sensori

- Nella parte destra della finestra real-time(F8) ci sono tre gruppi di bottoni.Rappresentano i Sensors il primo, i Controllers i secondi e gli Actuators i terzi.Questi sono i mattoni logici del GameEngine di Blender (LogicBricks).Nella parte superiore i tre gruppi sono uguali.Contengono tre bottoni ciascuno.

### **Sel**

-Se selezionato i LogicBricks degli oggetti selezionati nella finestra 3D sono visibili.

### **Act**

-Se selezionato i LogicBricks dell'oggetto attivo sono visibili

### **Link**

-Se selezionato i LogicBricks degli oggetti connessi al Controller sono visibili

-Nella parte bassa invece i tre gruppi cambiano.Contengono un pulsante grande con il nome dell'oggetto attivo ed uno Add il quale aggiunge altri sotto ogni gruppo.Il primo aggiunge un Sensor,vale a dire un evento,il secondo un Controller vale a dire il modo di controllare l'evento dall'oggetto ed infine il terzo un Actuator ,in pratica l'azione che succede. I Sensor hanno una piccola sfera nella parte esterna destra.Serve per connettere il sensore con l'anello nella parte esterna sinistra dei Controller, i quali contengono una sfera nella parte esterna destra la quale si connette con l'anello degli Actuators situati nella parte esterna sinistra.

---

## **Sensors**

-Premendo sopra il bottone Add del primo gruppo a destra nella finestra real-time(F8) abbiamo detto si aggiunge un sensor. I sensors sono come dei sensi,riescono a catturare un evento.Gli venti possono essere collisioni,input dalla tastiera, movimento del mouse,presenze vicine, ecc..Per ogni sensor in fondo metteremo anche i metodi dell'uso con la programmazione in Python. Vediamoli uno per uno:

---

### **Always**

- Questo è il sensore di che si apre sempre dopo avere premuto Add.Per cambiarlo bisogna tenere premuto il pulsante sinistro del mouse sopra Always e nella finestra pop-up scorrere il mouse dove si desidera.
- Questo sensor anche dal suo nome si capisce che un sensore che agisce sempre.Vale a dire, se un Controller è connesso a questo Sensor e poi a sua volta connesso ad un Actuator,l'azione che l'Actuator contiene succederà per tutta la durata del gioco.Avrete notato che il Sensor Always contiene diversi campi nel suo interno.
- Nella parte sinistra in alto il pulsante contrassegnato dalla x cancella questo sensore.
- Andando verso destra c'è il pulsante del tipo di questo Sensor(nel nostro caso,Always)
- L'altro campo più a destra è il nome del Sensor. E buona pratica mettere un nome diverso da quello di default(sensor,sensor1).Nelle scene complesse sarà più facile districarsi fra i LogicBricks.
- Il quarto pulsante con la freccia serve per comprimere o espandere questo Sensor.
- I tre bottoni sotto sono le pulsazioni del sensor.Le pulsazioni hanno due valori Vero(il primo pulsante) e Falso(il secondo pulsante).Quando un Controller è connesso ad un Sensor ,avvia l'Actuator ogni volta prende un impulso e l'Actuator a sua volta diventa attivo quando la pulsazione è vera e si disattiva quando è falsa.Facciamo un esempio.Se con il mouse dobbiamo passare sopra ad un oggetto per animarlo,avendo premuto il primo pulsante(Vero) quando il mouse passa sopra l'oggetto si anima, e quando il mouse e altrove si ferma.Nel caso il secondo pulsante fosse premuto

succederebbe il contrario. Il terzo pulsante determina la frequenza delle pulsazioni. Si misurano in frame per secondo (se il numero impostato è 50 significa che il Sensor si avvia ogni secondo).

-Più a destra c'è un altro pulsante nominato Inv, il quale inverte l'output dell' Sensor, in pratica il Pulsante pulsazioni Vero diventa Falso e viceversa. Nel caso nessuno di questi pulsanti fosse premuto il Sensor agisce una volta sola all'inizio del gioco.

-Questi pulsanti sono uguali per tutti i Sensor

---

### **Il sensore della Tastiera (Keyboard Sensor)**

-Premendo su Always e tenendo premuto nella finestra che appare si sceglie Keyboard abbiamo selezionato il sensore della tastiera. Questo sensore prende l'input dalla tastiera. Premere sopra il campo accanto alla scritta Key apparirà la scritta Press any key. Premete qualche tasto sulla tastiera ed il gioco è fatto. Il tasto premuto apparirà scritto sul campo. Il campo dopo, AllKeys, se selezionato fa sì che il sensore si avvii da tutti i tasti della tastiera. Sotto, dove c'è la scritta Hold si possono inserire altri due tasti modificatori, i quali vanno tenuti premuti mentre si preme il tasto principale. Il sensore della tastiera si può usare anche semplicemente per inserire del testo. Creiamo una proprietà di tipo String, li diamo un nome e lo inseriamo nel campo Target. Ora, finché un'altra proprietà qualsiasi inserita nel campo LogToggle è TRUE (vero) premendo i tasti della tastiera si inserisce del testo che viene salvato nella proprietà inserita nel campo Target

Metodo python:

**setKey** (chiave di tipo intero)

-Inserisce il tasto principale

**getKey()**

-Estrae il valore del tasto principale

**setHold1** (chiave di tipo intero)

-Inserisce il primo modificatore

**getHold1()**

-Estrae il valore del primo modificatore

**setHold2** (chiave di tipo intero)

-Inserisce il secondo modificatore

**getHold2()**

-Estrae il valore del secondo modificatore

**getPressedKeys()**

-Estrae il valore del tasto premuto

**getCurrentlyPressedKeys()**

-Estrae il valore del tasto che si tiene premuto

---

### **Il Sensore del Mouse (Mouse Sensor)**

-Questo sensore cattura il mouse, il suo movimento, la pressione sui tasti ed anche se si trova sopra un oggetto.

Clickando sul tasto lungo si può scegliere l'opzione che si desidera fra:

Left button click sul tasto sinistro del mouse

Middle button click sul tasto centrale del mouse

Right button click sul tasto destro del mouse

Movement si avvia ad ogni movimento del mouse

Mouse over si avvia quando il mouse si trova sopra l'oggetto

Metodi python:

**getXPosition()**

-Estrae il valore della coordinata X della posizione del mouse

**getYPosition()**

-Estrae il valore della coordinata Y della posizione del mouse

---

### **Il sensore del tatto (Touch Sensor)**

-Questo sensore si avvia quando l'oggetto viene in contatto con un materiale. Se nel campo MA: si inserisce il nome di un materiale il sensore reagisce solamente a quel materiale. Se non viene indicato nessun materiale reagirà a tutti i materiali.

Metodo python:

**setProperty** (nome materiale)

-Inserisce il nome del materiale al quale l'oggetto deve reagire

**getProperty()**

-Estrae il nome del materiale al quale l'oggetto ha reagito

**getHitObject()**

-Estrae l'oggetto con il quale, è avvenuto il contatto

### ***getHitObjectList()***

-Estrae una lista degli oggetti con i quali è avvenuto il contatto

---

### **Il sensore della collisione(Collision Sensor)**

-Questo sensore dà un impulso quando succede una collisione fra l'oggetto attivo e un altro oggetto nel ambiente circostante. Se si vuole limitare la individuazione della collisione non a tutti gli oggetti circostanti, nel campo Property: si deve inserire il nome di una proprietà. Allora si individuerà solamente la collisione con gli oggetti che hanno questa proprietà. La stessa procedura è valida se al posto di una proprietà si vuole inserire il nome di un materiale. Premendo sopra M/P si cambia il campo accanto da Property: a Material:

#### Metodo python:

***setProperty***(nomeproperty/nomemateriale)

-Inserisce il nome della proprietà o del materiale con il quale si deve individuare la collisione

***getProperty***()

-Estrae il nome della proprietà o del materiale con il quale c'è stata una collisione

***getHitObject***()

-Estrae l'oggetto con il quale, è avvenuta la collisione

***getHitObjectList***()

-Estrae una lista degli oggetti con i quali è avvenuta la collisione

---

### **Il sensore della vicinanza(Near Sensor)**

-Questo sensore invia un impulso ogni volta che nelle vicinanze trova un altro oggetto. L'area nella quale gli altri oggetti(solamente gli oggetti Actor) sono cercati si delimita da una sfera con il raggio pari al valore nel campo Dist. Questo sensore dà due impulsi, uno quando l'oggetto entra nell'area sferica con raggio il valore Dist e uno quando esce dall'area sferica con il valore Reset. Facciamo un esempio: Se abbiamo una porta la quale si apre quando l'oggetto attore entra nell'area sferica con raggio Dist si chiude quando l'oggetto attore esce dall'area Reset.

Anche per questo sensore come per gli altri due sopra se si vuole delimitare il campo dell'individuazione degli oggetti intorno si può inserire il nome di una proprietà nel campo Property: in modo da reagire solamente agli oggetti che hanno questa proprietà.

#### Metodo python:

***setProperty***(nomeproperty)

-Inserisce il nome della proprietà dell'oggetto il quale si deve individuare

***getProperty***()

-Estrae il nome della proprietà il quale è stato individuato

***getHitObject***()

-Estrae l'oggetto il quale, è stato individuato

***getHitObjectList***()

-Estrae una lista degli oggetti i quali sono stati individuati

---

### **Il sensore radar(Radar Sensor)**

-Questo sensore come si nota anche dal nome individua degli oggetti in area conica definita dalla distanza in Dist, da un angolo inserito nel campo Ang: ed anche dalla direzione determinata da uno dei bottoni X(asse positiva x), Y(asse positiva y) e Z (asse positiva z). Se nel campo Prop è inserito il nome di una proprietà il radar individuerà solamente gli oggetti che hanno questa proprietà.

#### Metodo python:

***setProperty***(nomeproperty)

-Inserisce il nome della proprietà dell'oggetto il quale si deve individuare

***getProperty***()

-Estrae il nome della proprietà il quale, è stato individuato

***getHitObject***()

-Estrae l'oggetto il quale, è stato individuato

***getHitObjectList***()

-Estrae una lista degli oggetti i quali sono stati individuati

---

### **Il sensore delle proprietà(Property Sensor)**

-Questo sensore controlla il valore delle proprietà. Esistono quattro modi per controllare questo valore. Il primo ed anche quello di default è:

Equal

-il sensore reagisce solamente se la proprietà nel campo Prop: ha un valore uguale al valore nel campo Value:  
Not Equal  
-il sensore reagisce solamente se la proprietà nel campo Prop: non ha un valore uguale al valore nel campo Value:  
Interval  
-il sensore reagisce solamente se la proprietà nel campo Prop: ha un valore che si trova nell'intervallo fra Min: e  
Max:  
Changed  
-il sensore reagisce solamente se la proprietà nel campo Prop: cambia valore  
Metodo python:  
**setProperty**(nomeproperty)  
-Inserisce il nome della proprietà da controllare  
**getProperty**()  
-Estrae il nome della proprietà controllata  
**setType**(modo di tipo intero)  
-Inserisce il tipo della proprietà. Il modo deve essere un intero come qui sotto:  
Equal = 1  
Not Equal = 2  
Interval = 3  
Changed = 4  
**setValue**(valore)  
-Inserisce il valore da controllare  
**getValue**()  
-Estrae il valore della proprietà

---

#### **Il sensore del raggio(Ray Sensor)**

- Emette un raggio con lunghezza il valore Range e direzione che si può scegliere nella finestra Type accanto a Range. Se un oggetto qualsiasi o uno che ha una proprietà con nome uguale a quella nel campo Property: o un materiale con nome uguale al materiale nel campo Material: (usate M/P per cambiare o Property: o Material:), si trova ad intercettare il raggio, il sensore da un impulso.

Metodo python:

**getHitPosition**()

-Estrae una lista che contiene la posizione dove il raggio colpisce l'oggetto

**getHitNormal**()

-Estrae una lista che contiene il vettore della normale come il raggio colpisce una superficie.

**getRayDirection**()

-Estrae una lista che contiene il vettore della direzione del raggio

**getHitObject**()

-Estrae l'oggetto il quale, è stato colpito

---

#### **Il sensore della generazione impulsi casuali(Random Sensor)**

-Questo sensore genera impulsi casuali che dipendono da valore di partenza Seed  
Per avere valori veramente casuali si devono impostare anche i pulsanti delle pulsazioni.

Metodo python:

**setSeed**(numero intero)

-Inserisce un numero come partenza del generatore

**getSeed**()

-Estrae il valore di partenza del generatore

---

#### **Il sensore dei messaggi(Message Sensor)**

-Se un messaggio arriva, il sensore emette un impulso. Il messaggio può essere filtrato in modo da ricevere solamente certi messaggi, impostando una stringa nel campo Subject:

Metodo python

**getBodies**()

-Estrae una lista con i corpi dei messaggi impostati nell'Actuator che manda i messaggi

**getFrameMessageCount**()

-Estrae il numero dei messaggi ricevuti dall'ultimo Frame

**setSubjectFilterText**()

-Inserisce il testo del messaggio che avvia il sensore

---

## 9.8 Controllori & Esecutori

### Controllori(Controllers)

-Nella parte centrale dei tre gruppi che si trovano alla sinistra della finestra real-time(F8)si trovano i Controllori.Dopo che l'evento viene catturato dai sensori la palla passa ai Controllori,che in pratica simboleggiano l'oggetto che controlla l'impulso arrivato dai Sensors e poi lo passa agli Actuators.Esistono quattro modi per controllare gli input che arrivano dai Sensors:

#### AND

-questo Controller ha la funzione di passare direttamente l'impulso come lo riceve.Notate bene però,se sono stati connessi due più Sensors il Controller tratta l'impulso secondo l'operazione logica And.In altre parole è vero se entrambi i Sensors sono veri.

#### OR

-questo Controller invece tratta l'impulso secondo l'operazione logica Or.Vale a dire che basta uno dei Sensors connessi a lui sia vero ed è vero.

#### EXPRESSION

-questo Controller ha un campo il quale può contenere un'espressione.Solamente nel caso si soddisfi la condizione posta dall'espressione il Controller passa, l'impulso. I tipi delle espressioni sono,Intero,Float,Boolean,String,Proprietà,NomeSensore.Le operazioni aritmetiche valide in una espressione sono,somma,sottrazione,divisione,moltiplicazione,maggiore di,maggiore o uguale di,minore di(+,-,/,\*,>,>=,<).Le operazioni logiche valide sono NOT,OR,AND,Uguale(==).

#### PYTHON

-Questo Controller rende il GameEgine molto flessibile e potente.Nel campo Script si può inserire il nome di un script in python dalla finestra Text(Shift + F11).

#### Metodo python:

**getActuator**(nomeactuator)

-Estrae l'Actuator con nome nomeactuator

**getActuators**()

-Estrae una lista degli Actuators connessi al controller.La lista ha indice iniziale zero.

**getSensor**(nomesensor)

-Estrae il sensore connesso al controller con nome nomesensor

**getSensors**()

-Estrae una lista dei sensori connessi al controller.La lista ha un indice iniziale zero

---

### Esecutori(Actuators)

-Gli actuators giocano il ruolo degli esecutori.Le azioni da eseguire li vediamo più in dettaglio qui sotto:

#### L'esecutore delle azioni(Action Actuator)

-Questo Actuator è valido solo per le armature.Serve per attivare un'animazione delle armature.(Per approfondire la conoscenza sulle armature [www.blenderchar.com](http://www.blenderchar.com).) Esistono cinque modi di eseguire l'animazione,i quali si possono attivare premendo con il mouse sopra il campo Play e poi scorrere il mouse sulla modalità desiderata.

#### Play

-Esegue l'azione che comincia dal frame Sta e finisce al frame End ad ogni impulso ricevuto

#### Flipper

-Esegue l'azione contenuta nei frame da Sta a End in base al impulso .Quando l'impulso arriva l'azione parte e quando l'impulso cessa, l'azione ritorna indietro fino al frame Sta anche se non arrivato al frame End.

#### Loop Stop

-Esegue l'azione in un ciclo finché l'impulso è positivo(l'azione da Sta aEnd).Se l'impulso diventa negativo si ferma al frame corrente

#### Loop End

-Esegue ripetutamente l'azione(l'azione da Sta aEnd) finché l'impulso è positivo.Quando smette l'azione continua fino al ultimo frame e poi si ferma.

#### Property

-Esegue l'azione nel frame determinato dal valore della proprietà nel campo Prop

#### L'esecutore del movimento(Motion Actuator)

-Questo è l'actuator più importante,perché muove,ruota,sposta l'oggetto.Accanto al campo che indica il tipo di movimento(Force,dLoc,dRrot ecc)ci sono tre pulsanti nei quali si può inserire il valore per ogni asse(x,y,z).Il quarto pulsante L se selezionato fa sì che il movimento si esegua nell'asse locale altrimenti i movimenti hanno come riferimento gli assi globali. L'Actuator inizia il movimento quando prende impulso positivo e si ferma quando l'impulso diventa negativo.Per muovere un oggetto per una certa distanza bisogna dare un impulso negative dopo ogni impulso

positive. Spieghiamo i tipi di movimento:

#### **Force**

-Una forza viene applicata agli oggetti dinamici.

#### **Torque**

-Una forza di rotazione viene applicata agli oggetti dinamici.

#### **dLoc**

-Sposta l'oggetto

#### **dRot**

-Ruota l'oggetto per un certo angolo

#### **linV**

-Imposta la velocità lineare dell'oggetto. Se si seleziona il pulsante add la velocità si somma a quella esistente

#### **angV**

-Imposta la velocità angolare.

#### Metodo python:

##### **setForce(list)**

-Inserisce il valore della forza da applicare sotto forma di una lista di tre elementi

##### **getForce()**

-Estrae la forza applicata all'oggetto sotto forma di una lista di tre elementi

##### **setTorque(list)**

-Inserisce il valore del campo Torque

##### **getTorque()**

-Estrae il valore della forza ruotante

##### **setDLoc(list)**

-Inserisce il valore del campo dLoc

##### **getDLoc()**

-Estrae il valore dLoc

##### **setDRot(list)**

-Inserisce il valore del campo dRot

##### **getDRot()**

-Estrae il valore dRot

##### **setLinearVelocity(list)**

-Inserisce il valore del campo linV

##### **getLinearVelocity()**

-Estrae il valore

##### **setAngularVelocity(list)**

-Inserisce il valore del campo angV

##### **getAngularVelocity()**

-Estrae il valore angV

#### **L'esecutore della limitazione(Constraint Actuator)**

-Questo Actuator limita il movimento dell'oggetto nella direzione desiderata(all'inizio è impostato su None) nella misura che si trova fra Min e Max. Volendo si può aggiungere della difficoltà di movimento in quella direzione impostando anche Damp diverso da zero.

#### Metodo python:

##### **setDamp(valore)**

-Inserisce un valore per Damp

##### **getDamp()**

-Estrae il valore attuale di Damp

##### **setMin(valore)**

-Imposta il valore di Min

##### **getMin()**

-Estrae il valore di Min

##### **setMax(valore)**

-Imposta il valore di Max

##### **getMax()**

-Estrae il valore di Max

##### **setLimit(tipo)**

-Inserisce un tipo per il limite(None=1,LocX=2,LocY=3,LocZ=4)

##### **getLimit()**

-Estrae il tipo di limite usato

#### **L'esecutore delle animazioni IPO(Ipo Actuator)**

-L'oggetto animato tramite le curve IPO si può eseguire in GameEngine tramite questo Actuator. Il GameEngine

sopporta solamente le IPO delle mesh(Loc,Rot,Size,Col),luci(Loc,Rot,RGB,Energy), Camera(Loc,Rot,Lens,ClipSta,ClipEnd). Se un altr'oggetto è stato imparentato con il nostro oggetto attivo e a sua volta contiene un IPO, attivando il bottone Child,viene eseguito anche quella animazione.Mentre il bottone Force converte le IPO che contengono spostamento (Loc)in una forza applicata all'oggetto, che se selezionate anche il bottone piccolo L viene eseguita sugli assi locali. I modi di eseguire queste animazioni sono:

#### **Play**

-Esegue l'animazione che comincia dal frame Sta e finisce al frame End ad ogni impulso ricevuto

#### **PingPong**

-Al primo impulso esegue l'animazione dall'inizio alla fine(Sta,End).Al secondo lo esegue al contrario(End,Sta)

#### **Flipper**

-Esegue l'animazione contenuta nei frame da Sta a End in base al impulso .Quando l'impulso arriva l'animazione parte e quando l'impulso cessa, ritorna indietro fino al frame Sta anche se non arrivato al frame End.

#### **Loop Stop**

-Esegue l'animazione in un ciclo finche l'impulso è positivo(l'azione da Sta aEnd).Se l'impulso diventa negativo si ferma al frame corrente

#### **Loop End**

-Esegue ripetutamente l'azione(l'azione da Sta aEnd) finche l'impulso è positivo.Quando smette l'animazione continua fino al ultimo frame e poi si ferma.

#### **Property**

-Esegue l'animazione nel frame determinato dal valore della proprietà nel campo Prop

#### Metodo python:

##### **SetType(tipo)**

-Imposta il tipo di animazione eseguita.(Play=1,PingPong=2 ecc)

##### **GetType()**

-Estrae il tipo di animazione eseguita(è u numero intero Play=1,..,Property=6)

##### **SetSta(frame)**

-Imposta il frame iniziale

##### **GetSta()**

-Estrae il frame iniziale

##### **SetEnd(frame)**

-Imposta il frame finale

##### **GetEnd()**

-Estrae il frame finale

#### **L'esecutore della camera(Camera Actuator)**

-Questo Actuator in pratica simula una camera di un gioco in terza persona.Nel campo OB: si deve inserire il nome del oggetto da seguire.Height è l'altezza della camera dall'oggetto.Min e Max impostano la distanza minimale e massimale della camera dall'oggetto.I bottoni X eY determinano a quale asse deve stare dietro la camera.

#### **L'esecutore del suono(Sound Actuator)**

-Questo Actuator esegue un suono da un file audio caricato nella finestra Sound Buttons . Una volta aggiunto premere sul bottone con il segno - e scorrere il mouse sul file audio desiderato(nel caso aveste caricato più di uno).Premendo il pulsante Custom set si aggiungono altri due pulsanti, il volume e l'altezza del suono. I suoni vengono eseguiti in questi modi:

##### **Play Stop**

-Esegue il suono finche l'impulso è positivo

##### **Play End**

-Esegue il suono fino alla fine quando un impulso positivo è stato dato.

##### **Loop Stop**

-Suona e ripete il suono quando impulso è stato dato.

##### **Loop End**

-Suono ripetutamente finche c'è un impulso.Quando non c'è più allora lo esegue fino alla fine e si ferma.

#### Metodi python:

##### **setGain(valore)**

-Imposta il valore del volume

##### **getGain()**

-Estrae il valore del volume

##### **setPitch(valore)**

-Imposta il valore dell'altezza del suono.

##### **getPitch()**

-Estrae il valore dell'altezza del suono

#### **L'esecutore delle proprietà(Property Actuator)**

-Questo Actuator serve per maneggiare le proprietà,assegnandoli,aggiungendoli dei valori o copiando il valore di

un'altra proprietà. Sarebbe consigliabile usare il python per eseguire queste operazioni, perché risparmierebbe della memoria. Quando è stato impostato su Assign, l'Actuator assegna il valore inserito nel campo Value alla proprietà inserita nel campo Prop, mentre se è stato impostato su Add, aggiunge il valore Value al proprietà nel campo Prop. Il terzo modo è Copy che copia il valore della proprietà Prop, la quale appartiene all'oggetto OB: nel ultima riga, nella proprietà Prop nella riga sotto Copy

#### Metodi python:

##### **setProperty(nome)**

-Inserisce il nome della proprietà che l'Actuator deve usare.

##### **getProperty()**

-Estrae il nome della proprietà che si sta usando

##### **setValue(valore)**

-Inserisce il valore della proprietà

##### **getValue()**

-Estrae il valore della proprietà

#### **L'esecutore degli oggetti(Edit Object Actuator)**

-Questo actuator compie operazioni sugli oggetti. Aggiunge altri (che devono essere in un layer non visibile) usando come fonte quello principale, sostituisce la mesh con un'altra, finisce la vita o lo ruota nella direzione di un altro oggetto. I quattro modi di esecuzione sono:

##### **Add**

-Aggiunge un altro oggetto OB: che attualmente si trova in un layer nascosto, avendo come origine la posizione dell'oggetto principale. Time decide il tempo in frame della vita dell'oggetto aggiunto. (0 vuol dire sempre). Nel linV si può inserire la velocità del movimento in una delle tre assi. L fa sì che questo movimento si esegua sull'asse locale.

##### **End Object**

-Finisce la vita dell'oggetto principale cancellandolo dal gioco.

##### **Replace Mesh**

-Sostituisce il nostro oggetto con un'altra mesh. Molto utile per tecniche di LOD (livelli di dettaglio)

##### **Track to**

-Gira l'asse Y dell'oggetto verso un altro indicato da OB: Time decide quanto velocemente avviene questo movimento. 3D se selezionato determina se questa rotazione viene fatta in tutte e tre le direzioni o solamente sul piano X/Y.

#### Metodo python:

-Per ogni modo soprannominato esistono vari metodi di programmazione in python.

##### **1) Per il modo di esecuzione Add**

##### **setProperty(nomeoggetto)**

-Inserisce il nome dell'oggetto da aggiungere. L'Actuator ne deve contenere uno, con il metodo si può sostituire.

##### **getProperty()**

-Estrae il nome dell'oggetto che si sta usando

##### **setTime(frames)**

-Inserisce il tempo l'oggetto aggiunto deve esistere (0=sempre)

##### **getTime**

-estrae il tempo dell'esistenza dell'oggetto aggiunto

##### **setLinearVelocity(list[vx,vy,vz])**

-Inserisce la velocità lineare del movimento dell'oggetto aggiunto

##### **getLinearVelocity()**

-Estrae la velocità lineare del movimento dell'oggetto aggiunto

##### **getLastCreatedObject()**

-Come si vede anche dal nome, estrae l'oggetto aggiunto per ultimo.

##### **2) Il modo End Object non ha metodi di programmazione in python.**

##### **3) Per il modo di esecuzione Replace Mesh**

##### **setMesh(nomeMesh)**

-Inserisce il nome di una mesh con la quale la mesh del nostro oggetto va sostituita. L'actuator ne deve contenere già uno.

##### **getMesh()**

-estrae il nome della mesh la quale deve sostituire l'oggetto

##### **4) Per il modo di esecuzione Track To**

##### **setProperty(nomeoggetto)**

-Inserisce il nome dell'oggetto verso il quale, il nostro va girato.

##### **getProperty()**

-Estrae il nome dell'oggetto verso il quale, il nostro va girato

##### **setTime(frames)**

-imposta il tempo della durata della rotazione

### ***getTime()***

-Estrae il tempo della durata della rotazione

**setUse3D(bool)**

-determina se la rotazione avviene(1) o no(0) in 3D

### **L'esecutore delle scene(Scene Actuator)**

-Nella barra del menu, accanto al campo SCE: che per default è 1, si trova un piccolo bottone con il segno meno. Premendo sopra si può scegliere se aggiungere un'altra scena al nostro gioco. È un buon metodo di creazione dividere i livelli in varie scene. Il nostro esecutore(Actuator) serve proprio per gestire questa scena. I modi d'uso sono:

#### **Restart**

-Ricomincia dall'inizio la scena attuale

#### **SetScene**

-Nel campo SCE inserisce il nome della scena da iniziare. Metodi di programmazione in python sono: **setScene(nomesцена)** e **getScene()**. Il primo imposta la scena da avviare, il secondo lo estrae.

#### **SetCamera**

-Questo modo è molto utile quando si usa più di una camera. Imposta il nome della camera che deve avere la visuale inserendo il suo nome nel campo Camera:. Metodi python sono: **setCamera(nomecamera)** e **getCamera()**

#### **AddOverlayScene**

-Aggiunge una scena sopra quell'attuale.

#### **AddBackgroundScene**

-Aggiunge una scena sullo sfondo.

#### **RemoveScene**

-Rimuove la scena attuale

#### **SuspendScene**

-Sospende una scena finché non è chiamato ResumeScene

#### **ResumeScene**

-Riprende la scena sospesa con SuspendScene

### **L'esecutore casuale(Random Actuator)**

-A volte si ha la necessità di generare dei valori casuali. Questo Actuator fornisce questa funzione, inserendo una proprietà nel campo Property ed un numero in quello Seed (il valore di start, altrimenti si riceve la stessa sequenza). I modi di funzionamento sono:

#### **Bool Constant**

-Si usa solamente per testare valori TRUE o FALSE

#### **Bool Uniform**

-Casualità 50%. True o False

#### **Bool Bernoulli**

-È uguale a Bool Uniform con l'aggiunta di un altro parametro, che è Chance. Questo parametro quando è per esempio impostato a 0.2 fa sì che il valore TRUE abbia il 20% delle possibilità di essere estratto.

#### **Int Constant**

-Testa un valore intero inserito nel campo Value:

#### **Int Uniform**

-Genera un numero casuale intero fra Min e Max

#### **Int Poisson**

-Genera dei numeri che hanno una media Mean la quale si raggiunge con un numero infinito di generazioni.

#### **Float Constant**

-Testa il valore float inserito nel campo Value

#### **Float Uniform**

-Genera un valore float tra Min e Max

#### **Float Negative**

-Genera un numero casuale float con una media Mean ed una deviazione SD

#### **Float Neg.Exp.**

-Questo valore casuale decade in modo esponenziale partendo dal valore iniziale Half-life time.

#### Metodi python:

#### **setSeed(valore)**

-Inserisce il valore iniziale del generatore

#### **getSeed()**

-Estrae il valore iniziale

#### **getPara1()**

-Estrae il primo valore del generatore casuale

#### **getPara2()**

-Estrae il secondo valore del generatore casuale

#### **setProperty(nome)**

-Inserisce il nome della proprietà da randomizzare

**getProperty()**

-Estrae il nome della proprietà

**setDistribution(intero)**

-Il modo in quale deve essere generata la casualità.(Bool Constant = 1,ecc..)

**getDistribution()**

-Estrae il modo che viene usato per generare valori casuali.Restituisce un valore intero(3 =Bool Bernoulli,ecc)

**L'esecutore dei messaggi(Message Actuator)**

-Questo esecutore manda dei messaggi che sono ricevuti dal sensore dei messaggi.Se nel campo To: viene specificato un nome di proprietà il messaggio viene mandato a quei oggetti che contengono questa proprietà, altrimenti viene mandato a tutti.Il soggetto va inserito nel campo Subject: Il contenuto Body può essere sia testo (T/P non selezionato) o Proprietà (T/P selezionato)

Metodi Python:

**setToPropName(nome)**

-Il nome della proprietà, che hanno gli oggetti ai quali li va mandato il messaggio.

**setSubject(stringa)**

-Il soggetto che si usa per filtrare i messaggi

**setBody(stringa)**

-Il contenuto del messaggio

**setBodyType(intero)**

-Determina se il contenuto è testo o proprietà

## 9.9 Python scripting per il GameEngine

-Il Python è un semplice ma abbastanza potente linguaggio object oriented([www.python.org](http://www.python.org)). In Blender si può usare per implementare funzioni non esistenti o migliorare quelle esistenti. Abbiamo visto che quasi ogni Logic Brick(Sensors(sensori), Controllers(controllori), Actuators(esecutori)) contiene qualche metodo di programmazione in Python. Come si usano?

-Nella finestra di testo(Shift + F11) premendo sul barra del menu l'icona con il segno meno -,si apre una finestrina di pop-up con due scelte(al momento), Add New e Open New. La prima aggiunge una nuova finestra di testo. Accanto al segno meno appare il nome dello script che stiamo per editare.Al momento si chiama Text.Per modificarlo basta premere sopra con il mouse e scrivere un altro nome. La seconda apre un file testo sul vostro Hard-Disk.Per fare sì che siano usate in Game Engine dobbiamo aggiungere sul Controller Python il nome dello script sulla finestra

-Il Game Engine fa uso di moduli interni già inclusi in Blender, i quali sono :

- 1) Il modulo GameLogic
- 2) Il modulo Rasterizer
- 3) Il modulo GameKeys

-Ognuno di questi moduli contiene dei metodi e funzioni.Il metodo più facile per visualizzare queste funzioni è questo: Create una mesh qualsiasi(un piano per es.), aggiungete un sensore Always al quale deselezioniamo il pulsante TRUE Mode e poi aggiungete un controller Python.

Aprirete la finestra di testo(Shift + F11) aggiungete una finestra nuova e digitate:

```
import Gamelogic
print dir(GameLogic)
```

se non avete cambiato il nome allo script, inserite nel controller Python il nome Text(se lo avete modificato inserite il nome che avete messo).Connettete i due (Sensor Always Controller Python) e con il cursore del mouse sopra la finestra 3D premete P(avviate il GameEngine).Nella finestra di debug(shell window) apparirà questa schermata:

```
['KX_CONSTRAINTACT_LOCX', 'KX_CONSTRAINTACT_LOCY',
'KX_CONSTRAINTACT_LOCZ', 'KX_CONSTRAINTACT_ROTXX',
'KX_CONSTRAINTACT_ROTYY', 'KX_CONSTRAINTACT_ROTZZ', 'KX_FALSE',
'KX_IPOACT_FLIPPER', 'KX_IPOACT_LOOPEND', 'KX_IPOACT_LOOPSTOP',
'KX_IPOACT_PINGPONG', 'KX_IPOACT_PLAY', 'KX_PROPSSENSOR_CHANGED',
'KX_PROPSSENSOR_EQUAL', 'KX_PROPSSENSOR_EXPRESSION',
'KX_PROPSSENSOR_INTERVAL', 'KX_PROPSSENSOR_NOTEQUAL',
'KX_RANDOMACT_BOOL_BERNOULLI', 'KX_RANDOMACT_BOOL_CONST',
'KX_RANDOMACT_BOOL_UNIFORM', 'KX_RANDOMACT_FLOAT_CONST',
'KX_RANDOMACT_FLOAT_NEGATIVE_EXPONENTIAL',
'KX_RANDOMACT_FLOAT_NORMAL', 'KX_RANDOMACT_FLOAT_UNIFORM',
```

```
'KX_RANDOMACT_INT_CONST', 'KX_RANDOMACT_INT_POISSON',  
'KX_RANDOMACT_INT_UNIFORM', 'KX_TRUE', '__doc__', '__name__', 'addActiveActuator',  
'error', 'getCurrentController', 'getRandomFloat', 'getSpectrum', 'setGravity', 'stopDSP']
```

-Vediamo in dettaglio quei metodi che c'interessano di più:

`getCurrentController()` estrae il Controller che contiene lo script

`addActiveActuator(nome,bool)` aggiunge un Actuator con nome. Mentre `bool=1` quando è attivo.

`getRandomFloat ()` Da un numero float casuale tra 0.0 e 1.0

`setGravity(list[gx,gy,gz])` imposta il valore della gravità

-Facendo la stessa cosa per gli altri due moduli si ottengono:

1) modulo Rasterizer

```
['__doc__', '__name__', 'enableVisibility', 'error', 'getWindowHeight', 'getWindowWidth',  
'makeScreenshot', 'setBackgroundColor', 'setMistColor', 'setMistEnd', 'setMistStart',  
'setMousePosition', 'showMouse']
```

-Vediamo qualcuno in dettaglio:

`enableVisibility(0 o 1)` rende gli oggetti visibili(1) o invisibili(0)

`getWindowHeight()` altezza della finestra 3D

`getWindowWidth()` larghezza della finestra

`makeScreenshot()` crea screenshot del game salvandoli come immagini .tga

`setBackgroundColor(r,g,b)` il colore del background in RGB

`setMistColor (r,g,b)` il colore della nebbia

`setMistStart(diststart),setMistEnd(distend)` la distanza da dove comincia e finisce la nebbia

`showMouse(0 o 1)` mostra(1) o no(0) il cursore del mouse

`setMousePosition(x,y,z)` posiziona il cursore del mouse

2) modulo GameKeys

-questo modulo definisce i nomi dei caratteri della tastiera come AKEY,BKEY,CKEY ecc.

-Metodi standard per i Sensors

`isPositive` il sensore è positivo

`getUsePosPulseMode()` se il True Pulse è selezionato è TRUE, se no è FALSE

`setUsePosPulseMode(bool)` se `bool` è TRUE seleziona True Pulse Mode, se è FALSE lo deselecta.

`getPosFrequency()` estrae la frequenza in True Pulse Mode

`setPosFrequency(int)` imposta la frequenza degli impulsi in True Mode

`getUseNegPulseMode()` - se il False Pulse è selezionato è TRUE, se no è FALSE

`setUseNegPulseMode(bool)` - se `bool` è TRUE seleziona False Pulse Mode, se è FALSE lo deselecta.

`getNegFrequency()` estrae la frequenza in False Pulse Mode

`setNegFrequency(int)` imposta la frequenza degli impulsi in False Mode

`getInvert()` guarda se il pulsante invert è selezionato o no

`setInvert(bool)` `bool` uguale True per selezionare Inv

-Cominciamo ad entrare un po' nelle tecniche di programmazione e di vedere qualche esempio che molte volte parla più di qualsiasi teoria. Il modulo che a noi c'interessa quasi il 90% è il modulo GameLogic. Le righe base di quasi qualsiasi script sono:

```
import GameLogic  
controller = GameLogic.getCurrentController()  
owner = controller.getOwner()
```

-la prima riga importa il modulo GameLogic, la seconda estrae il controllore dello script, e la terza il possessore dello script, vale a dire l'oggetto che subisce lo script. Il controllore è molto utile nella gestione dei LogicBricks, mentre il possessore è molto utile nel gestire l'oggetto stesso (posizione, orientamento, mesh ecc...). Se alle tre righe di sopra aggiungiamo anche:

```
print dir(owner)
```

-otterremo nella finestra di debug una schermata con i metodi del nostro oggetto:

```
['applyImpulse', 'disableRigidBody', 'enableRigidBody', 'getLinearVelocity', 'getMass',  
'getMesh', 'getOrientation', 'getParent', 'getPosition', 'getReactionForce', 'getVelocity',  
'restoreDynamics', 'setOrientation', 'setPosition', 'setVisible', 'suspendDynamics']
```

-Vediamoli un po' più in dettaglio

applyImpulse- applica un impulso all'oggetto  
 disableRigidBody disabilita il comportamento Rigid Body  
 enableRigidBody attiva il comportamento Rigid Body  
 getLinearVelocity estrae una lista di tre elementi che rappresentano la velocità lineare  
 getMass estrae la massa dell'oggetto  
 getMesh estrae la mesh dell'oggetto. Vedremo in avanti qualche script per spiegarlo in dettaglio.  
 getOrientation è una matrice 3x3 che rappresenta la rotazione dell'oggetto nello spazio  
 getPosition è una lista di tre elementi che rappresenta la posizione dell'oggetto  
 getVelocity - è una lista di tre elementi che rappresenta la velocità dell'oggetto  
 suspendDynamics sospende il comportamento dinamico degli oggetti che sono stati selezionati come Dynamic  
 restoreDynamics ristora il comportamento dinamico quando è stato usato suspendDynamics  
 setOrientation ruota l'oggetto nello spazio secondo una matrice 3x3  
 setPosition - posiziona l'oggetto  
 setVisible(bool) rende visibile(bool=1) o invisibile(bool=0) l'oggetto  
 -Cominciamo a vedere qualche script.

### 1) La posizione dell'oggetto:

```

import GameLogic
    controller = GameLogic.getCurrentController()
    owner = controller.getOwner()
    pos = owner.getPosition()
    pos[0] = pos[0] + 2
    pos[1] = pos[1] + 3
pos[2] = pos[2] + 4
owner.setPosition(pos)
  
```

-pos, è una lista di tre elementi(pos[0] = x, pos[1] = y, pos[2] = z) che rappresentano la posizione nello spazio(assi globali) dell'oggetto. Dopo aver aggiunto alla posizione x, 2 unità, y, 3 unità, z, 4 unità, portiamo il nostro oggetto nella posizione nuova con owner.setPosition(pos)

### 2) Il movimento di un oggetto

-In questo script imparando a muovere un oggetto, s'insegna anche come si aggiunge un Actuator con il Python. Collegiamo un Keyboard Sensor(freccia su) con il nostro Controller che contiene lo script e quest'ultimo a sua volta con un Motion Actuator con il nome act. Lo script è:

```

import GameLogic
    controller = GameLogic.getCurrentController()
    owner = controller.getOwner()
    actuator = controller.getActuator(act)
    actuator.setForce(50,0,0,1)
    GameLogic.addActiveActuator(actuator,1)
  
```

-allora nella quarta riga si estrae l'Actuator con nome act, poi nella quinta si imposta una forza nella direzione X uguale a 50 (vedere i metodi python del MotionActuator). Il quarto numero(1) significa che la forza si esercita tenendo presente gli assi locali. Nella sesta riga aggiungiamo l'actuator.

### 3) Cambiare le proprietà con python

-Le proprietà dell'oggetto, è meno dispendioso cambiarle con il python che usando i propri LogicBricks. Vediamo ora come estrarre da una proprietà di tipo Timer un cronografo che ci segnala i minuti e i secondi ed assegnare questo valore ad una proprietà di tipo Text (che con i font bitmap si può visualizzare sullo schermo). L'oggetto necessita di una proprietà di tipo Timer chiamata Time ed una proprietà di tipo string chiamata Text. Collegiamo un Always Sensor al Python Controller:

```

import GameLogic
import math
    controller = GameLogic.getCurrentController()
    owner = controller.getOwner()
    min = int(owner.Time / 60)
    second = math.fmod(owner.Time, 60)
    owner.Text = str(min) + : + str(int(second))
  
```

-nella seconda riga si introduce un altro concetto della programmazione in GameEngine (anche nel Blender)

Creator) che è, l'importazione dei moduli esterni avendo una installazione del Python(2.0 preferibilmente). Math è un modulo del python che come si vede anche dal nome contiene funzioni matematiche. Nella quinta riga possiamo vedere che le proprietà si possono assegnare direttamente al possessore(owner). Allora sempre nella quinta riga abbiamo assegnato alla variabile min il valore intero(int) della divisione della proprietà Time con 60. Il timer di Blender funziona in secondi. Nella riga di sotto(sesta) usiamo un metodo del modulo importato math, ossia fmod. Nell'ultima riga assegniamo alla proprietà Text, il valore dei minuti(min) e secondi(second) facendo le dovute conversioni e regolando l'output.

#### 4) La velocità di un oggetto

-Collegiamo un Always Sensor al Python Controller:

```
import GameLogic
import math
    controller = GameLogic.getCurrentController()
    owner = controller.getOwner()
    speed = owner.getVelocity()
    vel = math.sqrt(math.pow(speed[0],2) + math.pow(speed[1],2) + math.pow(speed[2],2))
```

-vel è la velocità dell'oggetto. Abbiamo usato altri due metodi del modulo math, pow e sqrt (elevazione a potenza e radice quadrata)

#### 5) Corsore del mouse

-Collegiamo un Mouse Sensor(chiamato mouse) attivando la modalità Mouse Movement con il Python Controller:

```
import GameLogic
    import Rasterizer
    ww = Rasterizer.getWindowWidth()
    wh = Rasterizer.getWindowHeight()
    aspect = float(wh)/ww
    actionx=28.0
    actiony=actionx*aspect
    c = GameLogic.getCurrentController()
    sensor = c.getSensor("mouse")
    owner = c.getOwner()
    owner.x = float(sensor.getXPosition()-ww/2)/ww*actionx
    owner.y = float(sensor.getYPosition()-wh/2)/wh*actiony
    owner.setPosition([owner.x,0,-owner.y])
```

-l'oggetto con il quale e collegato lo script(che serve da cursore) necessita di due proprietà float nominate x e y.

#### 6) Creare sequenza di immagini tga(screenshot)

-In Blender è possibile creare sequenza di immagini in formato tga anche durante lo svolgimento del gioco.

```
filebase="image"
import Rasterizer
import GameLogic
owner=GameLogic.getCurrentController().getOwner()
owner.nr=owner.nr+1
filename=filebase+"%d"%owner.nr+".tga"
print "Writing : "+filename
Rasterizer.makeScreenshot(filename)
```

-nel directory di Blender si salveranno delle immagini (image001.tga,image002.tga..) finche non si preme il pulsante Esc. L'oggetto necessita di una proprietà int nominata nr

#### 7) Controllo del volume del suono

-Dopo aver caricato un file audio ed averlo impostato in real-time in modo che il suono venga eseguito,allo stesso Sensor ,nominato Sound si può collegare un Python Controller che contiene lo script :

```
import GameLogic
cont = GameLogic.getCurrentController()
me = cont.getOwner()
sound = cont.getActuator("Sound")
```

```
sound.setGain(1.0)
```

-l'ultima riga controlla il volume(da 0.0 a 1.0)

### 8)Azioni con le mesh

-Abbiamo visto sopra una funzione del possessore chiamata getMesh.Facendo un esplorazione con `print dir(owner.getMesh())` si ottiene :

```
['getMaterialName', 'getNumMaterials', 'getTextureName', 'getVertex',  
'getVertexArrayLength']
```

-quella che a noi interessa di più è getVertex(). Perciò continuiamo ad esplorare anche questa funzione con `print dir(owner.getMesh().getVertex(0,0))` ed otteniamo:

```
['getNormal', 'getRGBA', 'getUV', 'getXYZ', 'setNormal', 'setRGBA', 'setUV', 'setXYZ']
```

-Le funzioni sono semplici da capire.Noi useremo solo due getXYZ ed setXYZ che praticamente estraggono ed impostano le coordinate dei vertici di una mesh.Collegando uno script come questo ad un piano che abbiamo suddiviso tre o quattro volte otterremo un effetto onda:

```
import GameLogic  
from math import *  
cont = GameLogic.getCurrentController()  
me = cont.getOwner()  
mesh = me.getMesh()  
vlen = mesh.getVertexArrayLength(0)  
fac = sin(me.timer*2.0)/2.0  
for vi in range(0,vlen):  
.....vert=mesh.getVertex(0,vi).getXYZ()  
.....vert[2]=(cos(vert[0]*3.0)+cos(vert[1]*2.0))*fac  
.....mesh.getVertex(0,vi).setXYZ(vert)  
-l'oggetto ha bisogno di una proprietà nominata timer di tipo Timer
```

---

## 10. Plugins

2002-08-21 02:18:06

Contenuto della sezione

---