Alma Mater Studiorum
Università di Bologna

SCUOLA DI INGEGNERIA E ARCHITETTURA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

Modellazione di Mesh tramite Operatori Booleani

Relatore:
Prof.ssa Serena Morigi

Tesi di Laurea di:
Vincenzo Moreno Luna

Anno Accademico 2013/2014 - Sessione III

Desidero innanzitutto ringraziare la prof.ssa Serena Morigi e il Dott. Flavio Bertini per tutto il supporto che mi hanno dato nel lavorare a questo progetto.
Ringrazio gli amici, i colleghi e i professori che hanno accompagnato tutto il mio percorso di studi, perchè da ognuno di loro ho imparato qualcosa che porterò sempre con me.
Ringrazio Costanza per avermi aiutato a correre in questa fase frenetica della mia vita.
Infine, voglio ringraziare la mia famiglia per avermi sempre sostenuto.

In particolare, ringrazio mia madre, a cui dedico questo lavoro.

Parole chiave: computer *grafica, modellazione geometrica, mesh poligonali, operazioni booleane, tangenza geometrica*

Indice

Introduzione

Nella computer grafica, nell'ambito della modellazione geometrica, si fa uso delle operazioni booleane tra solidi per la manipolazione e la creazione di nuovi oggetti. Queste operazioni, quali unione, intersezione e differenza, vengono applicate alle superfici degli oggetti 3D esattamente come si fa su altri insiemi. In questo modo si riescono ad ottenere nuove forme complesse come combinazione delle altre, che sono in genere più semplici.
Esistono diverse implementazioni per le operazioni Booleane, ed è possibile classificarle sulla base di tre diverse proprietà: il tipo di dati in input, il tipo di elaborazione e il tipo di dati in output.
Per quanto riguarda il tipo di elaborazione, questa può avvenire direttamente, utilizzando le superfici di partenza, o indirettamente, utilizzando rappresentazioni alternative per i dati di input. Riguardo al tipo di dati, esistono diverse tipologie di rappresentazione per gli oggetti 3D: basati su superfici Non-uniform Rational B-spline (NURBS), su superfici analitiche, mesh poligonali e altri ancora. La mesh poligonale è il metodo più utilizzato in Computer Graphics. In particolare, la mesh triangolare è la rappresentazione più versatile tra le mesh poligonali.
Un'operazione Booleana tra modelli poligonali può presentare diversi problemi. Molti sistemi CAD (Computer Aided Design) commerciali hanno difficoltà nel risolvere delle operazioni su modelli molto complessi. Può quindi capitare che il risultato prodotto non sia corretto, o che non sia prodotto alcun risultato. Alcuni sistemi non sono in grado di risolvere operazioni Booleane tra solidi di media complessità (100k-200k triangoli).
I problemi principali sono la stabilità numerica e il consumo di memoria, ma anche la performance può essere un problema. Ciò che pesa maggiormente sulla performance di un algoritmo per il calcolo delle operazioni Booleane tra due mesh è il calcolo delle intersezioni a cui esse danno luogo. Per quanto riguarda il consumo di memoria invece, questo aumenta all'aumentare del numero di facce di una mesh: l'uso di strutture dati efficienti per ospitare le informazioni relative agli elementi costitutivi di una mesh è fondamentale.
Un'applicazione che lavora con dati mesh poligonali utilizza generalmente una struttura dati derivata dalla winged- edge, calcolando le intersezioni dei lati. In genere, si usa una struttura gerarchica per classificare le entità geometriche. In questo caso, se il solido ha

un grande numero di facce, si possono presentare problemi di scarse performance e di memoria insufficiente. Altri sistemi evitano questi inconvenienti rappresentando le superfici mesh in forma implicita, ed effettuando le operazioni Booleane tra superfici implicite, per poi ritornare alla superficie risultante tramite estrazione della isosuperficie. Altri framework invece forniscono risultati approssimati usando il direct rendering o eseguendo operazioni locali solo sulle zone di intersezione. Questi sistemi forniscono solo risultati approssimati per massimizzare la performance. Inoltre, tendono a soffrire di problemi aggiuntivi.

Ciò che è stato realizzato in questo lavoro di tesi si colloca all'interno di un progetto preesistente, realizzato per consentire la manipolazione di modelli tridimensionali mediante l'utilizzo di operatori booleani: Mesh Glue [1].
Mesh Glue è un progetto sviluppato nell'ambito di un lavoro di tesi triennale, presso l'università di Bologna, dal Dott. Flavio Bertini, e consiste in un applicativo in grado di gestire mesh non strutturate, al fine di poterle utilizzare in ambienti in cui si applicano algoritmi di Subdivision Surface.
Le mesh ottenute mediante i normali software di modellazione o attraverso scansione di oggetti reali si rivelano in genere inadeguate per lo scopo, poichè troppo ricche di elementi e raffinate secondo criteri arbitrari. L'idea che sta alla base del progetto Mesh Glue è quella di iniziare a lavorare con oggetti semplici, che necessitano eventualmente di correzioni strutturali, per poi provvedere ad esempio al loro assemblaggio, ottenendo in questo modo delle buone mesh a cui applicare schemi di suddivisione. Questo dovrebbe mettere gli algoritmi di Subdivision Surface nelle condizioni ottimali per raffinare con precisione gli oggetti solo dove serve, e ottenere, come risultato, delle mesh libere dalle imposizioni originarie [1].
In questo lavoro, si è estesa la logica dell'applicazione degli operatori booleani, presente in Mesh Glue, per poter gestire anche scenari con mesh che presentano facce in tangenza. Inoltre, si è inserito Mesh Glue all'interno di un progetto più grande: Mesh Craft.
Mesh Craft è un progetto sviluppato nell'ambito di un lavoro di tesi di dottorato, presso l'università di Bologna, dallo stesso autore di Mesh Glue. Tale progetto consiste in un ambiente di modellazione che utilizza come sistema di input il Leap Motion Controller, un dispositivo capace di identificare le dita di una mano e seguirne i movimenti con alta precisione.
Come si vedrà, esistono vari prodotti, commerciali e non commerciali, che permettono di eseguire le operazioni Booleane tra Mesh e alcuni di essi forniscono un supporto per il Leap Motion Controller.
L'estensione di Mesh Craft sviluppata in questo lavoro verrà presentata con numerosi esempi e alla fine verrà confrontata con un prodotto non commerciale molto noto.

Capitolo 1

Mesh e Operazioni Booleane

In questa sezione vengono illustrati alcuni concetti base necessari per la comprensione del resto della trattazione: le mesh poligonali, le operazioni Booleane e le superfici di suddivisione.
A seguire, vengono presentati alcuni dei principali software di modellazione 3D che supportano le operazioni Booleane.

1.1 Mesh poligonali

Un oggetto 3D può generalmente essere rappresentato mediante una mesh poligonale. Una mesh è una rappresentazione approssimata di un oggetto, corrispondente a una tassellazione poligonale della sua superficie bidimensionale (fig. 1.1).
Questo tipo di rappresentazione consiste in una collezione di vertici, lati e facce (V, L, F) che definiscono la forma di un oggetto nella computer grafica. Le facce consistono solitamente di triangoli o quadrilateri, dato che questo semplifica il rendering, ma possono anche essere utilizzati altri tipi di poligoni convessi.

1.1.1 Topologia

La caratteristica fondamentale di una mesh è la connessione tra gli elementi che la definiscono: le facce poligonali, i loro lati e i loro vertici. Questi ultimi sono inoltre caratterizzati da una valenza, pari al numero di lati che vi incidono (fig. 1.1).

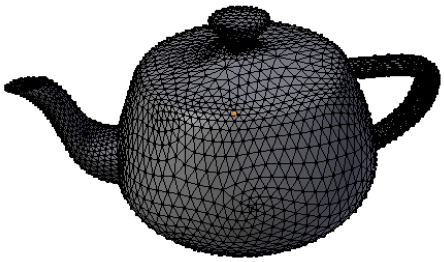


Fig. 1.1 - Mesh di un oggetto comune

Riguardo a questi rapporti di interconnessione, vi sono delle regole che una mesh deve rispettare per essere una rappresentazione corretta dell'oggetto approssimato. Tali regole sono:

- l'intersezione tra due facce della mesh può essere vuota, o avvenire in un lato o in un vertice comune;
- ogni lato appartiene ad almeno una faccia;
- ogni vertice appartiene ad almeno due lati.

Oltre a queste regole, è fondamentale che il contorno dell'oggetto sia una varietà bidimensionale (Two-manifold), ovvero che ogni punto della superficie abbia un intorno topologicamente equivalente (omeomorfo) ad un disco piano (esiste una biezione continua tra l'intorno ed il disco piano, per i punti di frontiera si richiede che il punto sia omeomorfo ad un semidisco). Quanto detto si traduce nel vincolo di appartenenza di ciascun lato ad al più due facce, e nella presenza, per ciascun vertice, di un ventaglio di facce adiacenti.
Un ultimo vincolo è legato alla rappresentazione grafica degli oggetti: l'orientamento dei poligoni. L'ordine ciclico dei vertici di una faccia ne determina l'orientamento e, per convenzione, l'ordinamento antiorario identifica il lato superiore della faccia. Se l'ordinamento dei vertici è compatibile per ogni coppia di facce adiacenti, ovvero se le facce adiacenti presentano i vertici del lato in comune in ordine inverso (fig. 1.2), la mesh risulta essere orientata.

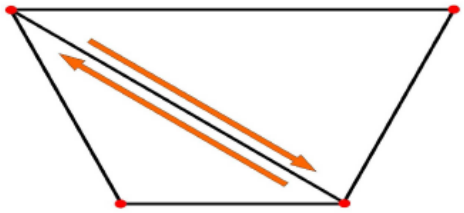


Fig. 1.2 - Orientamento di due facce adiacenti

Un'altra caratteristica di interesse per l'oggetto mesh è la possibilità di essere aperta, ovvero di possedere dei lati su cui incide un unica faccia, oppure essere chiusa e

presentare dei “buchi”. Si può definire formalmente questo secondo caso mediante la formula di Eulero:

$$V - L + F = 2 - G$$

dove V, L, F sono rispettivamente le cardinalità degli insiemi dei vertici, dei lati e delle facce della mesh. Con G (Genere) si intende il numero di curve chiuse lungo i quali si può tagliare la superficie senza disconnetterla; per una sfera, ad esempio. G = 0, mentre per un toro G = 1 (fig. 1.3).

Fig. 1.3 - A sinistra una sfera rappresentata con mesh poligonali, a destra un toro nella medesima rappresentazione.

Per concludere questa panoramica sull’oggetto mesh, si può dire che si possono distinguere due categorie principali di mesh: le mesh strutturate e le mesh non strutturate. Le mesh del primo tipo sono quelle in cui le facce e i vertici sono conformi fra loro, ovvero per ogni vertice e per ogni faccia si ha la stessa valenza, mentre quelle del secondo tipo presentano elementi con valenze eterogenee (fig. 1.4). L’eterogeneità presente nelle mesh non strutturate è il principale punto di forza di questo tipo di mesh durante la modellazione: la presenza di vertici a differente valenza, che consentono un migliore adattamento nei punti di curvatura della superficie che tassellano, le rendono maggiormente flessibili rispetto alle mesh strutturate [1].

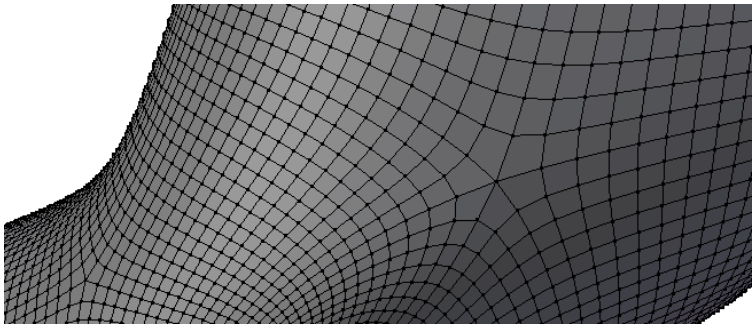


Fig. 1.4 - Dettaglio della superficie di una mesh non strutturata.

1.1.2 Struttura dati per mesh poligonale

Tipicamente l'oggetto mesh deve essere memorizzato su file. Vi sono diversi formati di file utilizzati per questa azione.
Il formato utilizzato in questo lavoro è uno dei più comuni: l'OBJ Wavefront. Un file in questo formato presenta una lista di punti 3D, corrispondenti ai vertici dell'oggetto mesh, seguita da una lista di n-uple, corrispondenti alle facce dell'oggetto definite dalla sequenza ordinata degli indici dei vertici nel file.
La capacità di avere una gestione efficiente delle informazioni che caratterizzano l'oggetto mesh durante il processo di modellazione dipende dalla scelta della struttura dati. Se si vuole soltanto rappresentare l'oggetto, le informazioni presenti in un file OBJ Wavefront sono sufficienti. Se invece si vogliono effettuare delle operazioni di editing della mesh servono strutture dati ben organizzate, volte a contenere informazioni addizionali, cercando un equilibrio tra la quantità di dati memorizzati e la rapidità di accesso a tali dati e il loro aggiornamento.
Una delle principali strutture dati presenti in letteratura è la Winged-Edge (fig. 1.5), che consente l'estrazione di informazioni derivate (informazioni aggiuntive ricavate dai dati contenuti nel file di memorizzazione) in tempi costanti. In questa struttura, i dati sono organizzati facendo perno sui lati: per ciascun lato si definiscono i suoi vertici (e con essi il verso di percorrenza, da X a Y), le due facce che vi incidono, e i lati predecessore e successore per ciascuna delle due facce incidenti. Per quanto le informazioni relative a vertici e facce, si memorizza per ogni entità un puntatore ad un lato incidente.

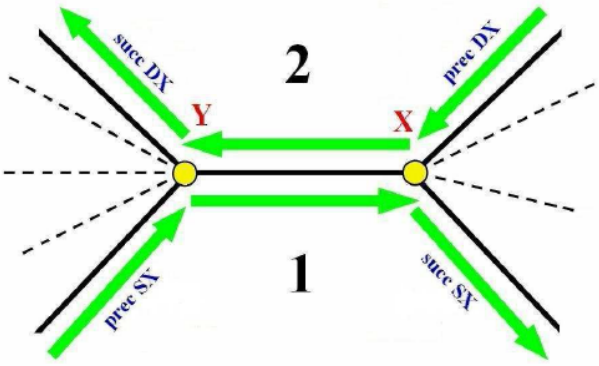


Fig. 1.5 - Modello Winged-Edge [1]

La struttura dati che si utilizza in questo lavoro si ispira fortemente a questo modello di memorizzazione. Infatti, questo tipo di struttura risulta particolarmente indicata nella gestione di mesh generiche perchè la disomogeneità delle mesh non strutturate richiede una gestione flessibile [1].

1.2 Operazioni Booleane tra mesh

Le operazioni Booleane (BO) tra solidi vengono utilizzate nella computer grafica per la manipolazione e la creazione di nuovi oggetti. Tali operazioni si applicano agli oggetti 3D al fine di ottenere delle nuove forme complesse a partire dalla combinazione di forme più semplici.
Dati due oggetti, vi possiamo applicare tre diverse operazioni: l'intersezione, individuata da un nuovo oggetto che include tutti i punti comuni ad entrambi gli oggetti originali; l'unione, individuata da un nuovo oggetto costituito da tutti i punti degli oggetti di partenza; la differenza, definita dall'insieme dei punti che appartengono a un oggetto e che non appartengono all'altro. A partire da due oggetti, vi sono naturalmente due possibili differenze (fig. 1.6).

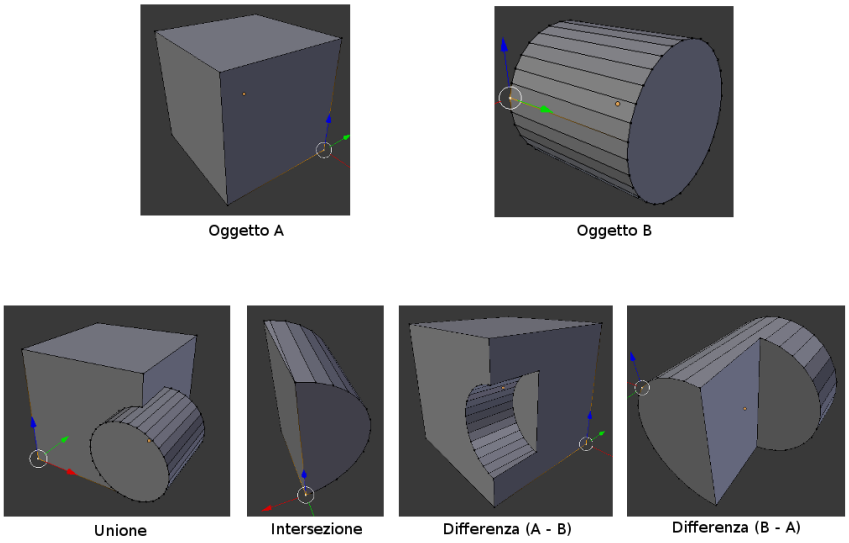


Fig. 1.6 - Operazioni Booleane tra mesh

In letteratura sono presenti diverse soluzioni per svolgere queste operazioni, ognuna con le sue peculiarità. Di seguito ne vengono presentate alcune.

Partendo dal presupposto che il costo di un operazione booleana è in genere dettato da tre passi (triangolarizzazione, classificazione inside/outside dei triangoli e le intersezioni tra le facce), Ming Chen, Xiao Yu Chen, Kai Tang e Matthew M. F. Yuen [4] hanno proposto un algoritmo per ridurre il costo della triangolarizzazione e della classificazione delle facce.
L'algoritmo proposto procede in quattro passi. Al primo passo si calcola l'intersezione dei bounding-box dei due oggetti. Se l'intersezione non è vuota, si calcola un modello LDI (Layered Depth Images) all'interno della regione di intersezione, con l'aiuto dell' hardware grafico che può decidere velocemente quali coppie di facce triangolari si intersecano tra loro. Al secondo passo, si calcolano i segmenti d'intersezione per ogni coppia di facce intersecanti e si collegano tali segmenti in poly-lines, le quali dividono

ogni faccia in diversi sub-domini che possono essere inside o outside. Al terzo passo, si classificano i sub-domini come inside/outside usando l'informazione del modello LDI che converte la classificazione da 3D a 1D, migliorando significativamente la performance dell'algoritmo. In seguito, la classificazione viene propagata a tutti gli altri triangoli/sub-domini non ancora classificati, completando la classificazione. Nell'ultimo passo, si triangolarizzano i sub-domini, ottenendo nuovi triangoli. I triangoli mantenuti (in base all'operatore booleano e alla classificazione dei triangoli), insieme a quelli appena generati, rappresentano il risultato finale.

F.R. Feito, C.J. Ogayar, R.J. Segura e M.L. Rivero [2] hanno proposto un algoritmo che opera direttamente sui triangoli della mesh e presenta tre passi principali. Il primo passo prevede l'intersezione dei due solidi coinvolti nell'operazione booleana. Il secondo passo esegue la classificazione dei triangoli ottenuti dal passo precedente. Il passo finale seleziona un insieme di
triangoli e costruisce un nuovo solido.
Per prima cosa, ogni solido viene scomposto in modo da ottenere un insieme di triangoli che soddisfa queste condizioni: ogni triangolo di un solido non deve intersecare un altro triangolo, e deve essere completamente dentro, completamente fuori o completamente sulla superficie dell'altro solido. In pratica, ogni triangolo di un oggetto viene intersecato con la superficie dell'altro solido, e si effettua una tassellazione ogni volta che si presenta un intersezione. Si adatta quindi la triangolazione di ciascun solido alle regioni di intersezione. A questo punto si classificano i triangoli in base alla loro posizione rispetto alla superficie dell'altro solido e, infine, si selezionano quei sottoinsiemi di triangoli che soddisfano certe condizioni per ciascun operatore booleano.
Gli obiettivi principali di questo algoritmo sono robustezza, precisione e rapidità nell'esecuzione.

Songgang Xu e John Keyser [3] hanno presentato una nuova tecnica per determinare le collisioni: si suddivide lo spazio in regioni cubiche mediante una struttura gerarchica, si costruisce una tabella hash per ospitare le primitive e riportare potenziali collisioni. Successivamente vengono individuate le collisioni sfruttando queste informazioni. Dopo aver determinato le collisioni, le informazioni raccolte vengono tradotte in un grafo.
Successivamente, si valutano le coordinate dei nuovi vertici e, sfruttando il grafo, li si connette creando nuove facce. Infine, si accoppiano i vertici dei due solidi (vertici originali e vertici nati dalle intersezioni) per arrivare a fondere le partizioni dei due solidi e generare il risultato finale.

Ancora, J.M. Smith e N.A. Dodgson [5] hanno sviluppato un algoritmo che può operare sia nel dominio 3D che nel dominio 2D. La parte principale dell'algoritmo si basa su una serie di operazioni interdipendenti che garantiscono la generazione di un risultato dotato di una corretta connettività. Le operazioni che vengono effettuate, che costituiscono il cuore di un operazione booleana, sono dei test che determinano, nel caso 3D, se un vertice di una struttura risiede all'interno dell'altra struttura, e se il lato di una struttura interseca una faccia dell'altra. Questi test si basano sui risultati di test equivalenti in un dominio 2D, in cui si assume che le strutture siano proiettate su un piano, in cui si verifica se un vertice risiede in una regione poligonale e se due lati proiettati si intersecano. I test 2D sono a loro volta basati su test effettuati in un dominio 1D, in cui si proiettano vertici e lati su di una linea. E' possibile lavorare sia con mesh triangolari che

con generiche mesh poligonali. L'algoritmo può generare alcuni artefatti geometrici, ma è possibile applicare una fase di post-elaborazione per rimuoverli.

1.3 Superfici di suddivisione

Al fine di avere un'idea riguardo l'uso effettivo delle mesh gestite in questo progetto, occorre definire il concetto di Subdivision Surface.
Come è stato detto, le mesh costituiscono una rappresentazione approssimata di un oggetto. La loro controparte è costituita dalle superfici matematiche esatte.
L'idea che sta alla base delle Subdivision Surface è quella di colmare il divario tra la rappresentazione approssimata e quella esatta, e per farlo si effettua un raffinamento della mesh originale, suddividendo le facce in poligoni sempre più piccoli, ottenendo al limite una superficie continua e regolare (fig. 1.7).
Uno dei vantaggi significativi di questa tecnica è la possibilità di limitare le procedure di raffinamento solo ai punti della mesh che richiedono una maggiore definizione, producendo delle mesh finali i cui poligoni si infittiscono solo nelle porzioni di superficie più critiche.
Anche se alcuni algoritmi di suddivisione sono in grado di regolarizzare dopo pochi passi di elaborazione la valenza delle facce, i vertici straordinari restano dei punti critici. Per capire meglio quali effetti possono avere questi punti, è necessario un richiamo di matematica: una funzione è di classe (o continuità) C^n se e derivabile sino all'ordine n-esimo e tutte le derivate sono continue; con C^{-1} si è in presenza di una discontinuità, con C^0 la funzione presenta un punto angoloso, con C^1 la tangente nel punto è continua e con C^2 è la curvatura ad essere continua.
Ritenere i punti straordinari dei punti critici, è dovuto al fatto che in loro corrispondenza le Subdivision Surface possono anche avere solo continuità C^0 o C^1 .
Il livello di continuità richiesto alla superficie dipende dall'ambito di applicazione: in alcuni può essere sufficiente il grado C^1, ma in altri può essere richiesto un grado di continuità almeno C^2.
Alla luce di quanto detto, risulta chiaro che mesh molto raffinate o con particolari errori strutturali non permettono agli algoritmi di Subdivision Surface di essere nelle condizioni ottimali di lavoro [1].

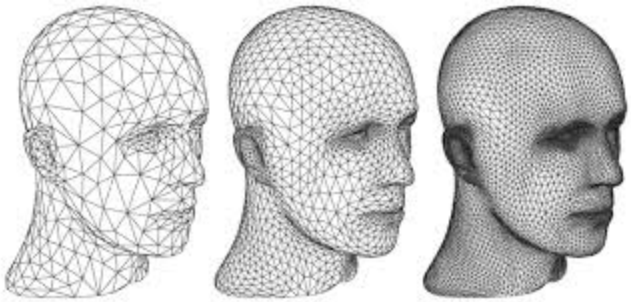


Fig. 1.7 - Raffinamento di una mesh mediante superfici di suddivisione

1.4 Soluzioni software esistenti per BO

Due pacchetti software commerciali per la modellazione 3D molto noti sono 3D Studio Max® (fig. 1.8) e Maya® (fig. 1.9), entrambi sviluppati da Autodesk®.

Fig. 1.8 - Autodesk 3D Studio Max®

Il primo dei prodotti menzionati offre una gamma di strumenti di modellazione molto robusti, con un enorme libreria di diversi modificatori che possono semplificare il lavoro di modellazione. Il secondo, invece, risulta un po più difficile da maneggiare ai fini della modellazione, sebbene abbia iniziato a offrire una serie di strumenti che semplificano il lavoro. Maya® mette inoltre a disposizione dell'utente un linguaggio di scripting che consente di creare strumenti di lavoro personalizzati.

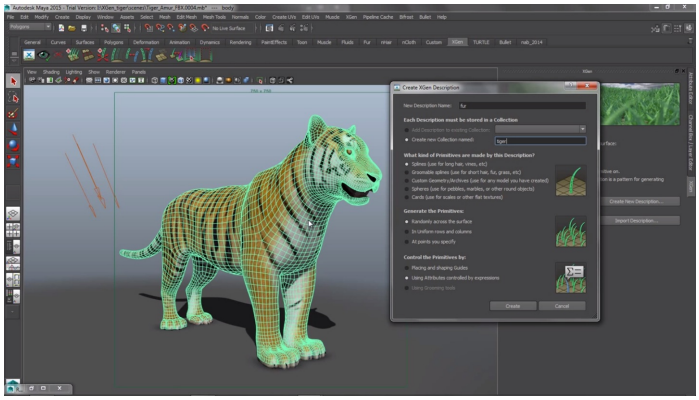


Fig. 1.9 - Autodesk Maya®

Entrambi i prodotti mettono a disposizione dell'utente la possibilità di effettuare operazioni Booleane tra mesh.

Un altro pacchetto software molto conosciuto, sempre sviluppato da Autodesk, è AutoCAD® (fig. 1.10). E' utilizzato principalmente per produrre disegni bi/tridimensionali in vari ambiti.
Anche AutoCad®, come i due prodotti precedenti, permette di effettuare BO tra mesh.

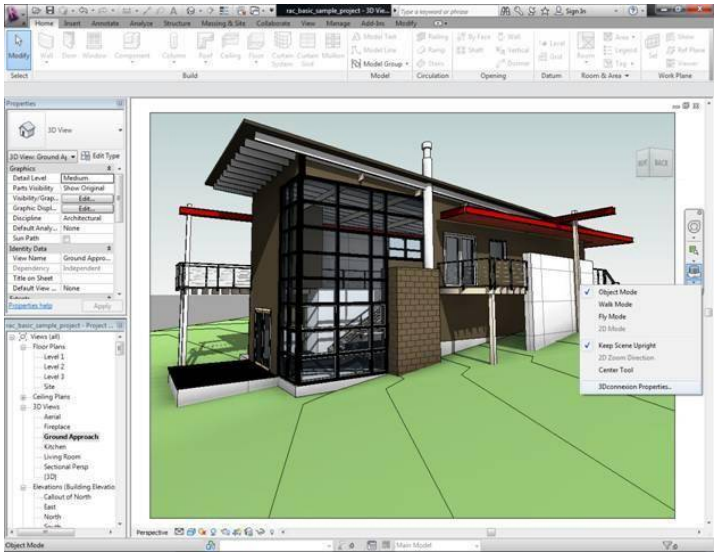


Fig. 1.10 - Autodesk AutoCAD®

Un altro applicativo per la modellazione 3D, che permette di effettuare operazioni Booleane tra mesh, è SketchUp® (fig. 1.11), software di modellazione 3D versatile, potente e semplice da utilizzare.
Una delle caratteristiche principali di SketchUp® è il motore di inferenza che assiste il disegnatore nella parte grafica. Esso permette un disegno molto accurato inferendo punti da altri punti, mentre si sta disegnando, e fornendo suggerimenti visivi che individuano punti significativi o condizioni geometriche.

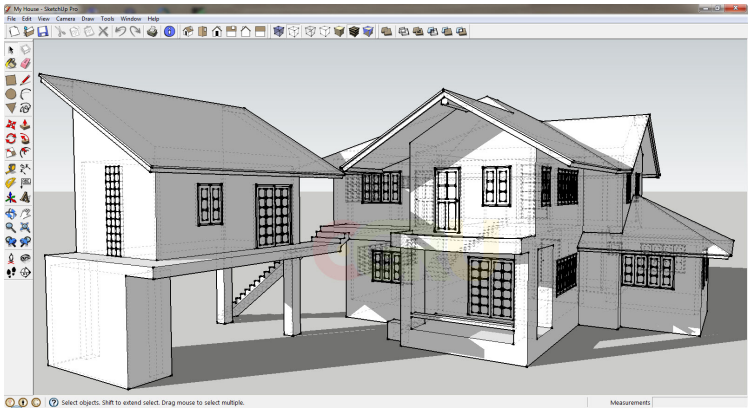


Fig. 1.11 - SketchUp®

Oltre ai software commerciali, esistono varie soluzioni anche nel mondo open source. L'applicativo libero maggiormente conosciuto è probabilmente Blender (fig. 1.12).
Blender è dotato di un robusto insieme di funzionalità, paragonabili, per caratteristiche e complessità, ad altri noti programmi per la modellazione 3D come 3D Studio Max e Maya. Naturalmente, anche questa soluzione offre, tra i suoi strumenti, la possibilità di effettuare le operazioni Booleane tra mesh.

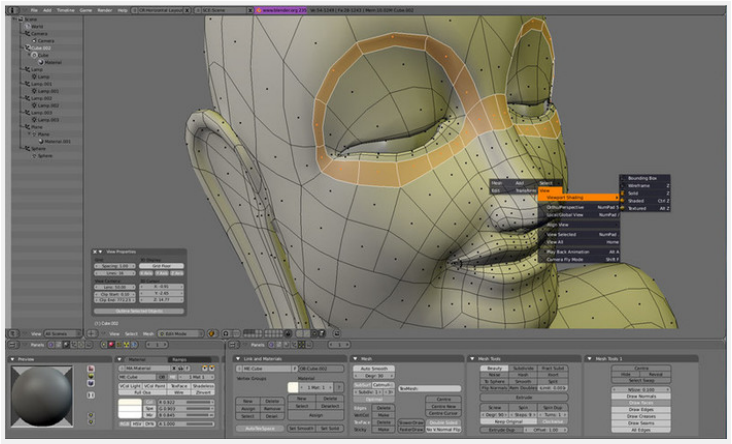


Fig. 1.12 - Blender

Capitolo 2

Il progetto Mesh Craft

Le capacità offerte dai software di modellazione vanno, naturalmente, incontro alle esigenze degli utenti. Tuttavia, quando l'esigenza degli artisti è legata alle modalità con cui si interagisce con il software di modellazione, diventa necessaria una soluzione di diversa natura.
Una novità abbastanza recente, che ha colpito il mondo della modellazione 3D, è stata quella del lancio sul mercato del Leap Motion Controller (LP).
Mesh Craft è un ambiente di modellazione, sviluppato nell'ambito di un lavoro di tesi di dottorato dal Dott. Flavio Bertini, che utilizza come sistema di input il Leap Motion Controller, prodotto dall'omonima azienda.
In questo capitolo viene fornita inizialmente una descrizione del LP Controller, e si prosegue con la presentazione del software Mesh Craft.

2.1 Leap Motion

La Leap Motion è una società specializzata nel fabbricare tecnologie di rilevazione del movimento per modificare l'interazione fra uomo e computer. La nascita di questa azienda è legata alla frustrazione che emerge nel processo di creazione di modelli 3D quando si utilizzano il mouse e la tastiera.
Il Leap Motion Controller è una piccola periferica USB (fig. 2.1) che, usando 2 telecamere e 3 LED infrarossi, osserva un'area approssimativamente a forma di semisfera di circa un metro. Il dispositivo è progettato per identificare dita, o oggetti simili (come una penna), con un altissima precisione.

Fig. 2.1 - Il Leap Motion Controller

Questo dispositivo ha ricevuto una calda accoglienza nel mondo della modellazione 3D: Autodesk ha subito sviluppato un plugin per consentirne l'utilizzo in Maya, ed è in corso l'integrazione con AutoCAD, mentre, nel mondo open source, c'è un primo supporto non ufficiale in Blender. Inoltre, nello store LP è presente un applicazione di modellazione (Freeform).
Il feedback degli utenti ha confermato che l'uso del controller permette di lavorare più velocemente di quanto non sia possibile utilizzando le tradizionali periferiche di input. Tuttavia, sono allo studio le gesture da utilizzare per ottimizzare l'esperienza con il Leap. La Leap Motion afferma che modellare creta virtuale con il suo controller è semplice come modellare la creta nel mondo reale.

2.2 Struttura di Mesh Craft

Il progetto Mesh Craft è strutturato nelle seguenti componenti:

- Gestione del controller
- Interfaccia grafica
- Oggetto mesh
- Engine geometrico
- Engine di supporto

Il componente per la gestione del controller si occupa dell'interfacciamento con il Leap Motion. Si appoggia direttamente a una libreria fornita dall'azienda produttrice.
L'interfaccia grafica mostra l'area di lavoro e permette di utilizzare tutte le funzionalità di Mesh Craft. Al momento, i menu che permettono di scegliere tra le varie azioni

disponibili, sono visualizzati mediante semplice testo e l'interazione con il menu avviene mediante tastiera (fig. 2.2).
Sia l'interfaccia grafica che la parte di gestione del controller non sono stati modificati in questo lavoro.

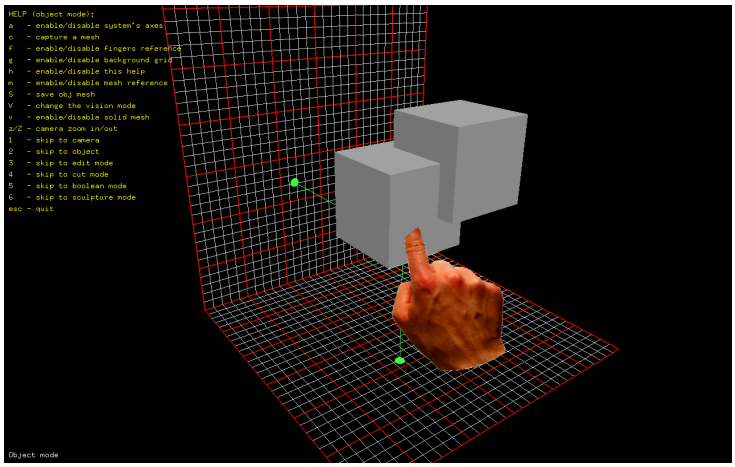


Fig. 2.2 - Area di lavoro in Mesh Craft

Il terzo componente è quello che gestisce gli oggetti tridimensionali: tutte le informazioni necessarie alla loro rappresentazione nell'area di lavoro, e quelle addizionali, necessarie alla loro manipolazione, si trovano qui.
La manipolazione degli oggetti avviene grazie alle funzionalità fornite dall'engine geometrico, il quale si apoggia a funzioni fornite dall'engine di supporto. In generale, si può dire che l'engine geometrico contiene tutte le funzioni strettamente legate all'oggetto mesh, mentre l'engine di supporto contiene funzioni matematiche e geometriche indipendenti.
Questo lavoro si è concentrato su questi ultimi tre componenti, in particolare sull'engine geometrico: è qui infatti che è stata inserita la funzione che si occupa di effettuare le operazioni Booleane tra mesh.Tale funzione, come anticipato nell'introduzione, è il frutto di un progetto precedente alla nascita di Mesh Craft: Mesh Glue.

2.2.1 Il progetto Mesh Glue

Viene illustrato adesso il funzionamento generale dell'algoritmo per le operazioni Booleane tra mesh presente nel progetto Mesh Glue. Le fasi dell'algoritmo classico che sono state oggetto di modifica in questo lavoro di tesi vengono esaminate con un maggior livello di dettaglio.

L'algoritmo presente in Mesh Glue opera in 6 fasi principali: calcolo delle intersezioni, costruzione del profilo di intersezione, inserimento del profilo di intersezione, split delle mesh, classificazione e generazione dei risultati.

La prima fase ha lo scopo di determinare, per entrambe le mesh, i punti di **intersezione** dei suoi lati con le facce dell'altro oggetto (fig. 2.3), tale ricerca viene effettuata controllando tutte le coppie lato faccia delle due mesh.

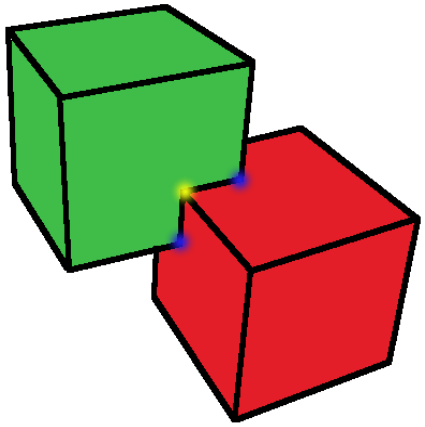


Fig. 2.3 - Due cubi si compenetrano in corrispondenza di uno spigolo: in blu alcune intersezioni generate dal cubo verde, in giallo alcune intersezioni generate dal cubo rosso.

La seconda fase, la più importante, è quella di **ricostruzione del profilo di intersezione**. Questa fase soffre di un'incertezza decisionale legata alla natura dei punti di intersezione. I punti individuati nella prima fase, infatti, possono essere di svariata natura: la faccia può essere intersecata internamente, sul perimetro o su un vertice, da un lato che a sua volta può generare l'intersezione in un estremo o al suo interno; queste molteplici combinazioni danno luogo a una incertezza proprio nell'ordinamento geometrico.
La procedura sviluppata procede nell'ordinamento in un alternanza tra i due insiemi di punti di intersezione generati dai lati delle due mesh, in modo da garantire sempre una scelta sicura.
La costruzione del profilo di intersezione deve iniziare da un punto che non provochi una situazione immediata di scelta ambigua, in particolare il punto di partenza non deve essere un vertice della mesh.
Da qui in poi, si procede iterativamente nella ricerca e i casi che si possono presentare sono essenzialmente due: la scelta del prossimo punto da inserire è sicura oppure è ambigua. La scelta sarà sicura se la faccia, relativa al lato a cui appartiene l'ultimo punto inserito, ha un numero di punti di intersezione interni minore di due, mentre sarà ambigua se il numero di punti interni alla faccia è maggiore di uno o se l'ultimo punto di intersezione inserito coincide con un vertice della faccia. Se si tratta di una scelta sicura potranno essere inseriti uno o due punti, due se la faccia ha un punto interno, mentre se la scelta risulta ambigua la ricerca si sposta sull'altra mesh, in particolare sulla faccia a cui appartiene il lato dell'ultimo punto inserito nel profilo (fig. 2.4).
Nella malaugurata ipotesi che un punto di intersezione sia generato dalla coincidenza di due vertici, allora la procedura non è in grado di decidere in sicurezza.

Se le mesh si intersecano in più di una regione, questa procedura viene ripetuta più volte per costruire tutti i vari profili di intersezione.

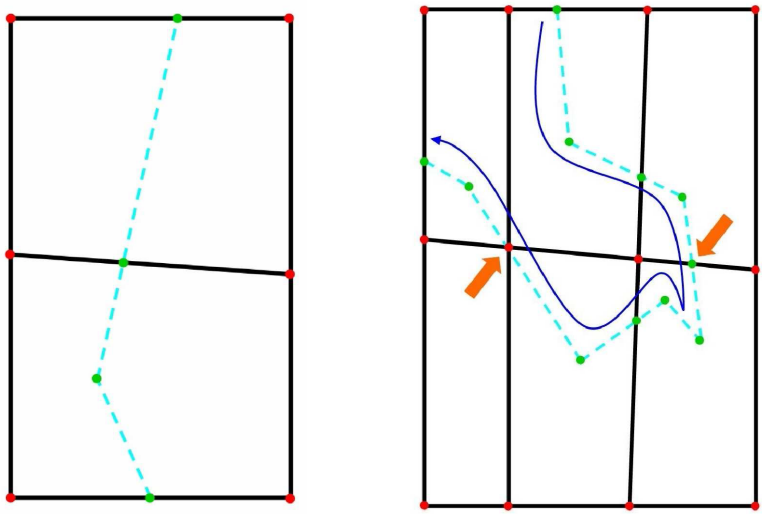


Fig. 2.4 - A sinistra un esempio in cui in ogni caso, la scelta del prossimo punto da inserire nel profilo è sicura (su qualunque lato si trovi il punto); a destra due esempi (indicati dalla freccia arancione) in cui la scelta del punto successivo, seguendo l'ordine definito dalla freccia blu, risulta ambigua [1].

La terza fase è quella di inserimento del profilo di intersezione nelle due mesh. Questa operazione provoca la **suddivisione delle facce attraversate dal profilo**. Naturalmente, viene applicata per ciascuno profilo individuato nella fase precedente.
Nell'operare sulle facce, si possono presentare tre diversi casi: la faccia non è interessata dal profilo, per cui verrà semplicemente ricopiata; un punto del profilo "tocca" il perimetro della faccia, alla quale sarà quindi aggiunto un vertice; ed infine, se si tratta di una faccia attraversata interamente dal profilo, questa dovrà essere separata in due, rispettando l'orientamento della faccia originale. In quest'ultimo caso la procedura presta particolare attenzione alla fase di separazione: se il punto di ingresso e il punto di uscita del profilo dalla faccia, appartengono a lati distinti, allora i vertici della prima faccia risultante saranno composti dai vertici compresi tra il punto di uscita e quello di ingresso seguiti dai vertici del profilo in ordine, mentre quelli della seconda faccia risultante saranno quelli compresi tra il punto di ingresso e quello di uscita e i vertici del profilo in ordine inverso. Al contrario, se il punto di ingresso e il punto di uscita appartengono allo stesso lato, a seconda dell'orientamento dei vertici originale, si produrrà una faccia risultante composta esclusivamente dai vertici del profilo. Per poter suddividere la faccia bisogna rilevare la porzione di profilo che la attraversa, in particolare si fa riferimento al punto di ingresso (dal quale si proseguirà sino al punto di uscita). Si può considerare tale solo un punto del profilo appartenente ad un lato della faccia, il cui successore nel profilo di intersezione appartenga esso stesso alla faccia (fig. 2.5).

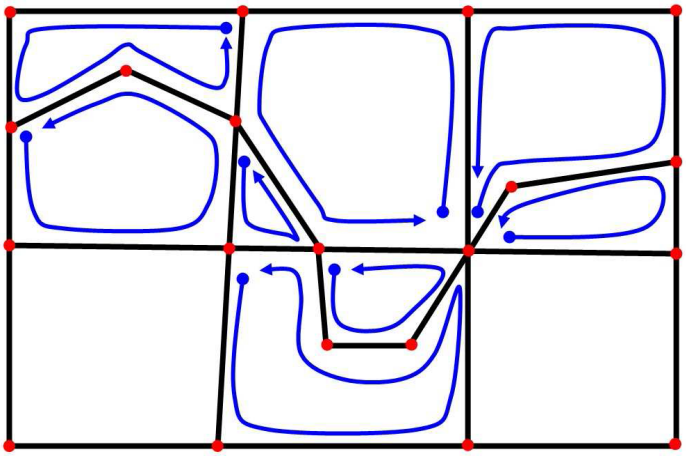


Fig. 2.5 - Esempio di come vengano separate le facce attraversate dal profilo nei diversi casi [1].

Nella quarta fase, **si separano effettivamente gli oggetti** con una procedura abbastanza semplice. Innanzitutto, si determina una faccia di partenza avente un lato coincidente con un segmento di un profilo di intersezione, e da questa si prosegue sulle facce adiacenti, individuate dai lati della faccia corrente che non appartengono ad alcun profilo (fig. 2.6). Con tutte le facce visitate in questo modo si genera una prima mesh (patch). Questa procedura viene ripetuta per ciascun profilo di intersezione, generando una nuova mesh ogni volta. Infine, si selezionano tutte le facce non appartenenti agli insiemi appena determinati per generare un'ultimo patch.

Fig. 2.6 - Esempio di come vengano identificate per ricorsione sui lati le facce appartenenti a uno dei due patch a partire da un lato [1].

Nella quinta fase, **si classificano i patch prodotti** nella fase precedente come appartenenti o non appartenenti all'intersezione tra le due mesh (interni o esterni all'altro oggetto). Per fare ciò, per ciascun patch, si prende un vertice che non appartiene ad alcun profilo di intersezione, e si considera una retta passante per questo punto e orientata in

una qualunque direzione: se questa retta interseca l'altra mesh (rispetto alla mesh di appartenenza del patch che si sta classificando) un numero dispari di volte, allora il punto del patch considerato è interno all'altra mesh; al contrario, se il numero di intersezioni è pari, il punto è considerato esterno all'altra mesh.

Nella sesta fase, si uniscono, secondo le modalità riportate in tabella, i patch prodotti sfruttando le informazioni di classificazione. Si **producono così quattro mesh risultato** per le due mesh di partenza, corrispondenti all'unione, all'intersezione, e alle due differenze.

Operazione	Patch di A utilizzati	Patch di B utilizzati
A ∪ B	Esterni a B	Esterni ad A
A ∩ B	Interni a B	Interni ad A
A - B	Esterni a B	Interni ad A
B - A	Interni a B	Esterni ad A

Capitolo 3

Gestione della tangenza geometrica

In questo capitolo viene trattato il cuore di questo lavoro: l'estensione della logica dell'algoritmo presentato nella sezione precedente al fine di riuscire a gestire correttamente situazioni di tangenza tra le facce delle due mesh.
L'ordine di presentazione degli argomenti segue la sequenza di operazioni dell'algoritmo e, dopo l'esposizione della logica, vengono illustrati aspetti aggiuntivi legati all'implementazione del codice.

3.1 Fase preliminare

Prima del calcolo delle intersezioni tra i due oggetti, si controllano tutte le facce appartenenti alla regione di intersezione delle due mesh, individuata da un bounding box, alla ricerca di facce che presentano una sovrapposizione parziale o totale, ovvero una zona di tangenza. Per fare questo, si effettua un test di complanarità tra le facce e, se l'esito è positivo, si procede con la ricerca dell'eventuale regione di sovrapposizione. Il test di complanarità consiste nella verifica dell'appartenenza di tre vertici di una faccia al piano in cui si trova l'altra faccia, dopo averne determinato l'equazione. La fase di ricerca della regione di sovrapposizione prevede invece di verificare l'esistenza di punti di intersezione tra i lati delle due facce, formare sequenze di lati in funzione dei punti individuati (fig. 3.1), classificare tali sequenze (interne o esterne all'altra faccia), e costruire la faccia di tangenza e le eventuali facce residue.

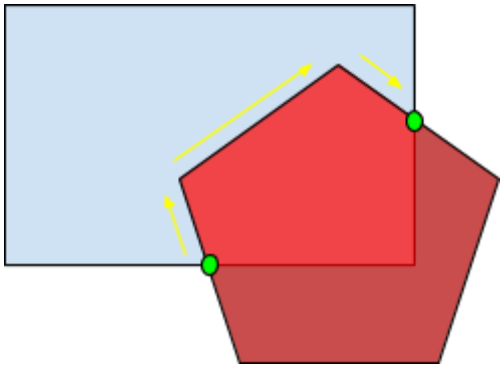


Fig. 3.1 - I punti di intersezione (in verde) generati dalle due facce individuano quattro sequenze di punti. Ciascuna sequenza inizia con un punto di intersezione e termina con un punto di intersezione. In giallo una delle quattro sequenze.

L'approccio utilizzato è logicamente identico a quello usato dall'algoritmo di Mesh Glue, ovvero: calcolo delle intersezioni, calcolo del profilo di intersezione, costruzione dei patch e classificazione.
Le zone di tangenza così individuate vengono inserite nelle facce originali, separando quindi la porzione di faccia tangente da quella non tangente, se presente (fig. 3.2).

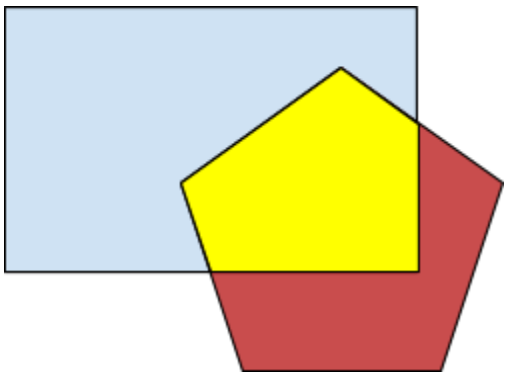


Fig. 3.2 - La faccia tangente (in giallo) viene inserita nella faccia rettangolare e nella faccia pentagonale, modificandone la forma. In questo esempio, ciascuna delle due facce originali viene sostituita da due nuove facce la cui combinazione corrisponde alla faccia originale.

Le facce tangenti e i loro vertici vengono poi marcati come interessati dalla tangenza.
A questo punto, è possibile procedere con il calcolo dei punti di intersezione tra le due mesh.

3.2 Costruzione del Profilo

L'idea alla base della logica di gestione delle facce tangenti è quella di mantenere l'approccio utilizzato nell'algoritmo classico, ovvero di determinare un profilo di

intersezione che attraversa le facce delle due mesh, anche in presenza di zone di tangenza. Tuttavia, un profilo di intersezione è costituito da una sequenza di punti, mentre le zone di tangenza sono costituite da facce. E' evidente che, volendo mantenere la rappresentazione del profilo di intersezione come una sequenza di punti, al fine di non modificare il comportamento dell'applicazione nelle fasi successive, è necessario ridurre le zone di tangenza alla medesima rappresentazione.
Per ottenere quanto appena detto, è sufficiente ricordare che le facce tangenti e i loro punti sono stati marcati come tali, e che ciascuna faccia tangente è stata ricavata da una faccia della mesh originale.
Quando, durante la costruzione del profilo, si valutano i punti tangenti, si considerano quei punti che, pur essendo dei vertici, sono situati su edge della mesh originale e dunque sono validi per l'inserimento.
Una volta aggiunto un punto di tangenza al profilo, si prende come riferimento la faccia non tangente su cui ci si sarebbe spostati se non ci fosse stata la tangenza (fig. 3.3), e si inseriscono tutti i punti che appartengono sia alla faccia tangente che alla faccia non tangente di riferimento.

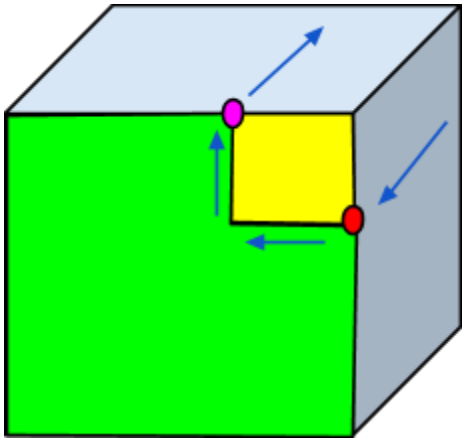


Fig. 3.3 - Quando, costruendo il profilo di intersezione, si arriva al punto tangente (indicato in rosso), si usa come guida la faccia non tangente (indicata in verde) per percorrere la sequenza di punti comuni a questa e alla faccia tangente (indicata in giallo), fino ad arrivare al punto tangente di uscita (in viola) situato su un edge della mesh originale. Da qui si continua con la normale costruzione del profilo.

L'ultimo punto tangente inserito, come il primo, è situato su un edge della mesh originale.
Si è così individuata la sequenza di punti che permette di attraversare la faccia nella mesh originale (fig. 3.3).
Adesso è possibile continuare la costruzione del profilo normalmente.
Nella situazione appena descritta, la zona tangente era costituita da un unica faccia tangente. Se invece la zona tangente è costituta non da una , ma da più facce tangenti, occorre verificare se l'ultimo punto inserito appartiene anche a un altra faccia tangente adiacente alla faccia di riferimento che si sta percorrendo (fig. 3.4). Il controllo viene ripetuto ad ogni inserimento e, se l'esito è positivo, ci si sposta sulla faccia tangente adiacente.

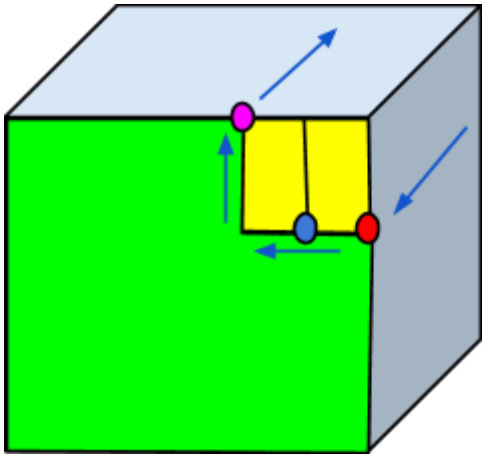


Fig. 3.4 - Quando si inserisce il punto blu, si verifica se è presente una faccia tangente adiacente. Se presente, ci si sposta su questa faccia e si continua con la costruzione del profilo.

Infine, se l'intersezione dei due oggetti è costituita da sole facce tangenti, ovvero non vi è un profilo di intersezione "canonico", è sufficiente non applicare la procedura. Infatti, le facce tangenti sono già state inserite e marcate come tali nella fase preliminare, e sono quindi pronte per essere inserite nel risultato, senza elaborazioni successive.

3.3 Altre fasi dell'algoritmo

In fase di inserimento del profilo, si considerano i punti tangenti solo se individuano l'attraversamento di una faccia non tangente (fig. 3.5).

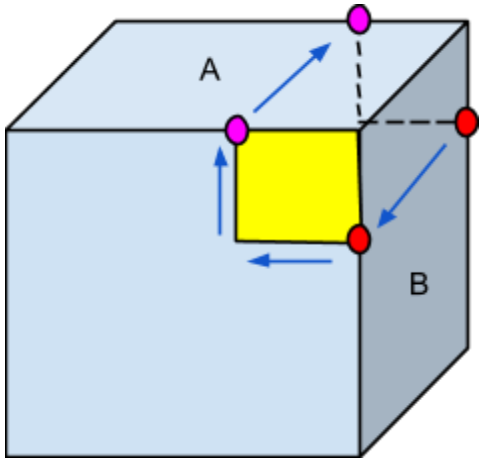


Fig. 3.5 - I punti tangenti viola provocano l'inserimento di un edge nella faccia A, suddividendola, mentre i punti tangenti rossi causano la suddivisione della faccia B. Le facce a cui appartiene sia un punto rosso che un punto viola non vengono modificate.

In fase di split, quando si compone un patch, adesso ci si ferma non solo quando si incontra un edge di profilo, ma anche quando si incontra un edge tangente. Le facce tangenti non vengono così incluse in nessuno dei patch prodotti.

Infine, nella fase di classificazione dei patch, la procedura rimane inalterata. Si aggiunge soltanto un'ulteriore operazione finale che prevede di inserire le facce tangenti, che appartengono ad entrambe le mesh, nei risultati di unione e intersezione. La costruzione dei risultati è sintetizzata in tabella.

Operazione	Patch di A utilizzati	Patch di B utilizzati	Facce tangenti
A ∪ B	Esterni a B	Esterni ad A	Si
A ∩ B	Interni a B	Interni ad A	Si
A - B	Esterni a B	Interni ad A	No
B - A	Interni a B	Esterni ad A	No

3.4 Implementazione del Codice

In questa sezione vengono illustrati dei dettagli aggiuntivi relativi al codice. Come nelle precedenti esposizioni, anche qui si segue l'ordine delle diverse fasi di elaborazione.
Inizialmente, verrà esposta la struttura generale secondo cui è organizzato il software e, in seguito, si illustreranno le strutture dati utilizzate.
Fatto questo, si esamineranno in dettaglio le varie funzioni che implementano le procedure descritte in precedenza.
Il linguaggio di programmazione utilizzato in Mesh Craft è il linguaggio C++.

Il progetto Mesh Craft è costituito principalmente da:

- CraftMesh_controller: gestore degli eventi generati dal controller
- CraftMesh_viewer: interfaccia grafica
- CraftMesh_ model: rappresentazione dell'oggetto mesh
- CraftMesh_gengine: motore geometrico
- Math: libreria di funzioni matematiche

Il progetto Mesh Glue è stato inserito in Mesh Craft sotto forma di una funzione principale, supportata da altre procedure, che permette a quest'ultimo l'applicazione degli operatori Booleani mediante invocazione.

Tutto il codice portato da Mesh Glue, e quello sviluppato nel corso di questo lavoro di tesi, è stato inserito in CraftMesh_gengine, CraftMesh_model e Math.

3.4.1 Strutture Dati

In questa sezione vengono illustrate le strutture dati più importanti del progetto, ovvero quelle relative alla memorizzazione di una mesh e quelle utilizzate per l'applicazione dei vari algoritmi. Come anticipato nel capitolo "Mesh e Operazioni Booleane", il formato di rappresentazione per le mesh di input/output che si utilizza in questo lavoro è l'OBJ Wavefront, mentre la struttura di memorizzazione è simile alla Winged Edge.

Tutte le informazioni di una mesh sono all'interno della classe ModSubdivMesh, che presenta tre sotto strutture, ciascuna delle quali legata ad uno degli elementi costitutivi della mesh (vertici, lati e facce).
Quando si carica una mesh da un file, le informazioni presenti nel file (vertici e facce) vengono inserite nelle apposite strutture. In seguito, richiamando la funzione *Mod_LoadSubdivMesh*, si producono ulteriori informazioni correlate e la lista dei lati della mesh, riempiendo completamente le strutture dati.
Le informazioni relative agli elementi costitutivi della mesh, inseriti nell'oggetto ModSubdivMesh, sono tutte quelle richieste dai vari algoritmi: valenza, adiacenza, posizione e appartenenza al bordo. Grazie all'organizzazione strutturata in classi di elementi, a loro volta costituiti da array, si garantisce una buona rapidità di accesso mantenendo la flessibilità richiesta dalle mesh non strutturate.

Oltre a queste informazioni, presenti in Mesh Glue e riportate in Mesh Craft nel corso della fusione dei due progetti, per gestire le situazioni oggetto di questo lavoro si è aggiunto a vertici, lati e facce l'informazione di tangenza. Ai vertici è stato inoltre associato un ulteriore flag di non appartenenza alla mesh originale.

Per quanto riguarda le altre informazioni presenti in Mesh Glue, relative all'esecuzione delle operazioni Booleane, queste sono state riportate invariate in Mesh Craft.
In particolare, i punti determinati nella fase di calcolo delle intersezioni vengono memorizzati in una coda single-link, annotandone, oltre alla coordinata di intersezione, gli elementi responsabili (lato, faccia), l'informazione di tangenza e quella di non appartenenza alla mesh originale. Inoltre, si fa uso di una coda double-link per la ricostruzione della poligonale di intersezione, facilitando così la ricerca di determinati punti nel momento in cui risulta necessario risalire il profilo ordinato fino a quel punto.
Per evitare un'inutile ridondanza di informazioni, la coda dedicata al profilo ordinato, contiene esclusivamente i puntatori ad elementi delle liste di punti di intersezione.

3.4.2 Dettagli di implementazione

Procedura di individuazione delle zone di tangenza

La procedura preliminare di gestione della tangenza è individuata dalla funzione *gen_fTg*, che riceve in input le due mesh e le restituisce dopo aver generato delle nuove facce, corrispondenti alle zone di tangenza tra le due mesh.

La prima operazione che viene fatta in questa funzione è il controllo di complanarità tra le facce delle due mesh. Per ridurre i tempi di elaborazione, tale controllo viene effettuato solo tra le facce che appartengono al bounding box del volume di intersezione: si verifica l'appartenenza di tre vertici di una faccia al piano in cui si trova l'altra faccia, dopo averne determinato l'equazione. Per ciascuna coppia di facce complanari, si procede con il calcolo della nuova faccia che individua la regione di sovrapposizione (se presente). Quest'ultima azione è definita dalla funzione *gen_faceOverlap* che, date le due facce e il piano su cui si trovano, restituisce la sequenza di punti che individua la zona di sovrapposizione e le sequenze di punti che individuano i residui delle facce originali una volta che si sia ritagliata da queste la zona di sovrapposizione (fig. 3.6).

Fig. 3.6 - La sovrapposizione di due facce individua una zona di tangenza e diverse aree residue per ciascuna faccia.

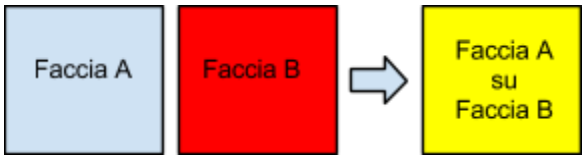
Per fare questo, la funzione *gen_faceOverlap* esegue diversi passi.

Innanzitutto, confronta tutti i lati di una faccia con tutti i lati dell'altra, alla ricerca di possibili sovrapposizioni totali o parziali tra gli edge. In presenza di sovrapposizione parziale, gli edge interessati vengono suddivisi in più edge, corrispondenti alla parte sovrapposta e alla parte non sovrapposta. Man mano che si percorrono gli edge, si compilano tre liste di edge: edge non sovrapposti per una faccia, edge non sovrapposti per l'altra faccia, edge sovrapposti. Le liste di edge non sovrapposti vengono utilizzate per determinare i punti di intersezione tra gli edge delle due facce.

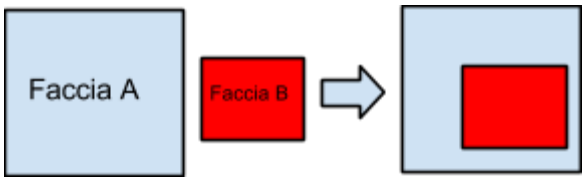
Prima di procedere al passo successivo, utilizzando la funzione *gen_pointDplane2D* si determinano per ciascuna faccia i punti interni all'altra: data una faccia convessa, per ogni suo vertice, si calcola la somma degli angoli tra il vertice e ogni coppia di vertici degli edge dell'altra faccia. Questa somma sarà pari a 2π se il punto si trova sul piano del poligono e al suo interno [9]; nel caso di faccia concava, si effettua una triangolarizzazione prima dell'applicazione del metodo.

A questo punto è possibile costruire le sequenze di punti che individuano le nuove facce. Vi sono quattro possibilità:

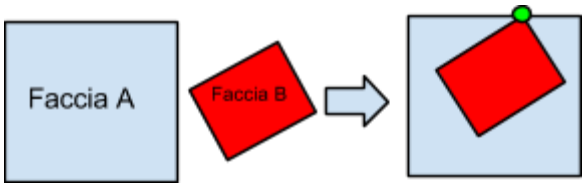
- tutti gli edge sono sovrapposti



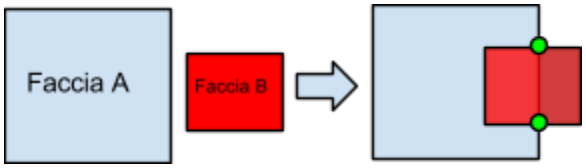
- non vi sono intersezioni



- vi è una sola intersezione



- vi è più di un'intersezione



Nel primo caso la gestione è molto semplice: le due facce si sovrappongono perfettamente, quindi la zona di tangenza coincide con le facce e non bisogna fare nulla. Nel secondo caso, invece, è necessario controllare se una faccia è interamente contenuta nell'altra, nel qual caso bisognerà inserirla: la faccia esterna viene suddivisa in tre facce (fig. 3.7).

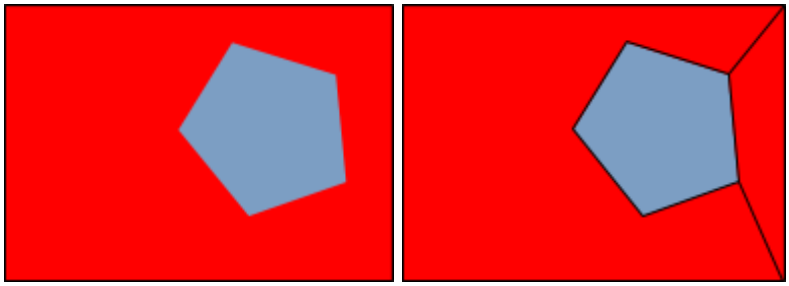


Fig. 3.7 - La sovrapposizione di una faccia, interamente contenuta nell'altra, causa la suddivisione della faccia esterna in tre facce.

Il caso di singola intersezione, sebbene tenuto presente in fase di analisi, attualmente non è gestito. Tuttavia, la politica di gestione da implementare è analoga a quella del caso precedente.
Il caso di intersezioni multiple è quello con la gestione più articolata: per entrambe le facce, si individuano le sequenze di punti comprese tra punti di intersezione (fig. 3.8), e si utilizzano per costruire la faccia di intersezione e le facce residue. In questa fase si utilizzano le liste di punti interni alle facce per determinare se una sequenza di punti appartiene all'intersezione dei due poligoni, e poter così effettuare le scelte opportune.

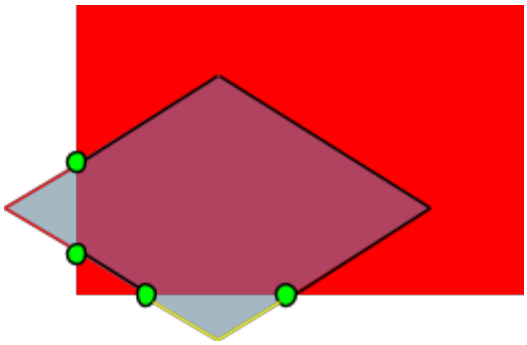


Fig. 3.8 - Sequenze di punti per la faccia romboidale

Le sequenze di punti non sono presenti solo nel caso di più intersezioni, anche nel caso precedente le nuove facce vengono rappresentate mediante sequenze di punti. In questo modo si ottiene uniformità nella gestione al passo successivo.

Una volta determinate tutte le sequenze, si passa all'operazione successiva di *gen_fTg*: l'inserimento dei profili di tangenza nelle facce, mediante la funzione *gen_insertTanProfile*. Per far questo, tutte le facce non interessate dalla tangenza vengono ricopiate, mentre quelle che presentano la tangenza vengono sostituite dalla faccia di intersezione e dalle faccette residue definite dalle sequenze di punti individuate prima.
Infine, si marcano le facce tangenti come tali, così come gli edge e i vertici che le costituiscono. Il risultato prodotto da *gen_fTg* consiste quindi in due mesh con sole facce

tangenti e non tangenti, dove le facce tangenti individuano le aree di contatto tra le due mesh (Fig. 3.9).

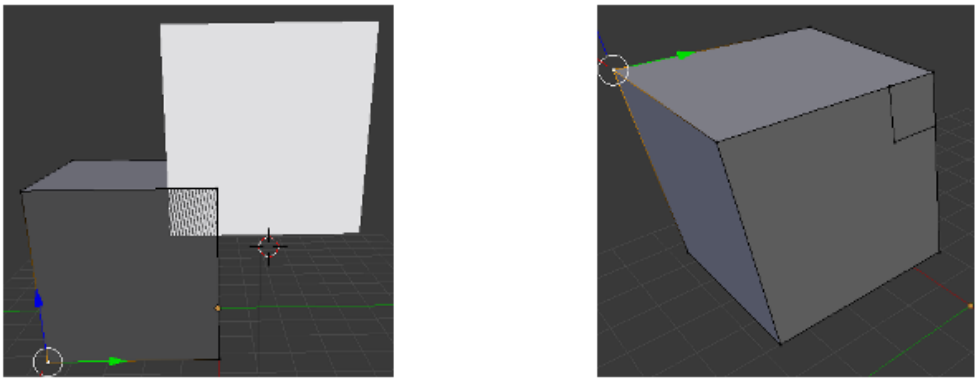


Fig. 3.9 - Le mesh di input (a sinistra) e una delle due mesh dopo l'invocazione di *gen_fTg* (a destra)

Procedura di costruzione del profilo

La funzione che si occupa di costruire il profilo di intersezione è la *gen_buildPolygonProfile*. Dato che il dettaglio della procedura è già stato trattato ampiamente nella descrizione dell'algoritmo di gestione della tangenza, qui si riportano soltanto dei dettagli aggiuntivi.
Innanzitutto, per gestire il caso in cui i punti di intersezione tra le due mesh sono tutti tangenti (fig. 3.10), è stato necessario modificare l'algoritmo per consentirgli di iniziare da un vertice di tangenza. Il principio di scelta del vertice di partenza è semplice: deve essere un punto posizionato su un edge della mesh originale, ovvero non doveva essere un vertice prima della fase di individuazione delle zone di tangenza.

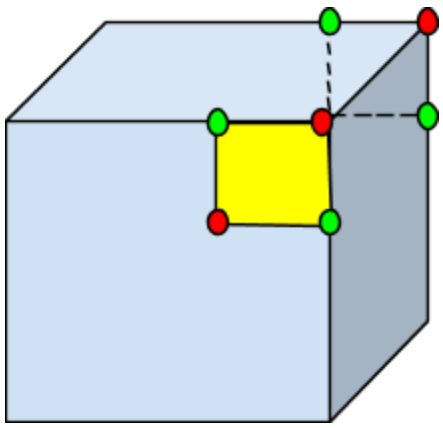


Fig. 3.10 - Cubo con due facce tangenti (in giallo e sul lato opposto del cubo). Solo i punti verdi possono essere utilizzati per iniziare la costruzione del profilo.

Prima di questo lavoro, man mano che si aggiungevano punti al profilo in costruzione, questi venivano rimossi dalle liste dei punti di intersezione fino a svuotarle del tutto. Con l'introduzione della logica di tangenza, quando si finisce di gestire una faccia tangente si rimuovono dalle liste delle intersezioni tutti i punti di questa faccia: a differenza dei punti di intersezione non tangenti, che vengono presi tutti, alcuni punti delle facce tangenti non vengono utilizzati (fig. 3.11).

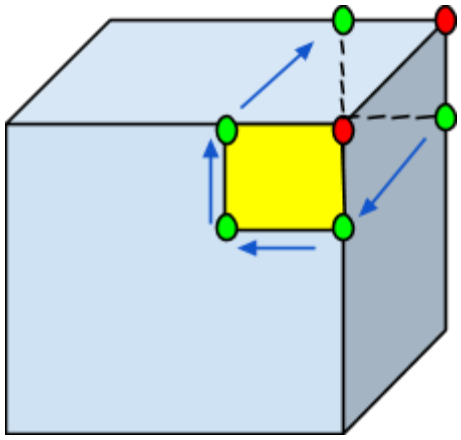


Fig. 3.11 - I punti rossi non vengono inseriti nel profilo, ma comunque rimossi dalle liste delle intersezioni una volta che siano state gestite le rispettive facce tangenti di appartenenza.

Altre funzioni

Le funzioni di inserimento del profilo (*gen_insertProfile*) e di split (*gen_splitMesh*) sono state oggetto di lievi modifiche e la logica che vi è stata aggiunta è stata già trattata nel paragrafo 4.4, così come la procedura di classificazione.

Capitolo 4

Sperimentazione

In questa sezione vengono presentati i risultati ottenuti sottoponendo a diverse prove le funzionalità aggiunte grazie al lavoro svolto e, per avere un termine di confronto, verranno mostrati i risultati ottenuti da Blender per gli stessi test. Vengono inoltre presentati due esempi nell'interfaccia grafica di Mesh Craft e due scenari non gestiti da questo lavoro.
Come nel lavoro originale [1], è doveroso sottolineare che la fase di sperimentazione non è da considerarsi conclusa con quanto verrà riportato in questa sezione. Sebbene infatti nella realizzazione di questo lavoro siano state esaminate e gestite diverse situazioni, non è da escludersi l'esistenza di altre situazioni non emerse in fase di analisi o in fase di test. Alla fine del capitolo vengono riportati i tempi di esecuzione per alcuni dei test presentati.

4.1 Test Mesh Craft

Test 1 (cubo1 cubo2)

La situazione creata in questo test consiste in due cubi che, intersecandosi, presentano due zone di tangenza (fig. 4.1).

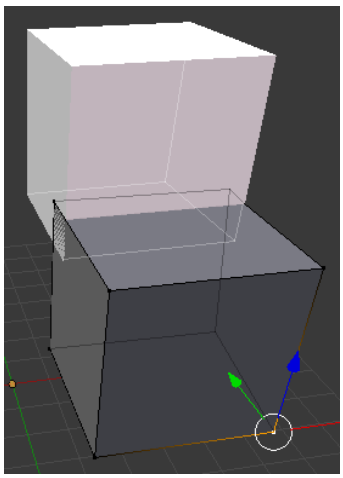


Fig. 4.1 - Due cubi si intersecano dando luogo a due zone di tangenza

Al termine della fase di gestione della tangenza, ciascuno dei due cubi presenta una nuova faccia, corrispondente alla zona di tangenza (fig. 4.2).

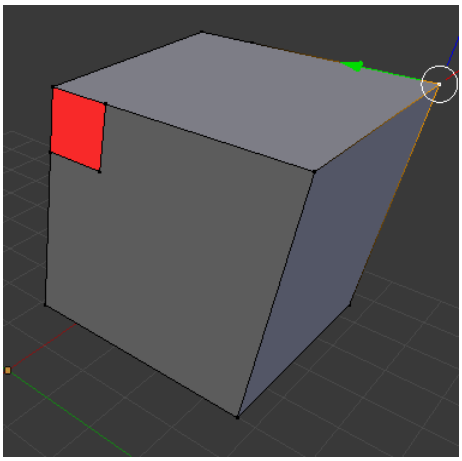


Fig. 4.2 - Le nuove facce individuano le zone di tangenza

A questo punto, si calcolano le intersezioni, si individua il profilo di intersezione e lo si inserisce in entrambe le mesh. In fig. 4.3 il risultato.

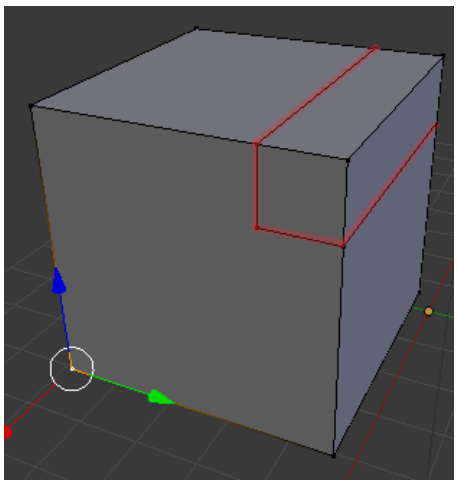


Fig. 4.3 - Uno dei due cubi dopo l'inserimento del profilo di intersezione

Dopo l'operazione di split e la classificazione dei patch, questi vengono combinati per generare i risultati di unione (fig. 4.4), intersezione (fig. 4.5), e differenza (fig. 4.6).

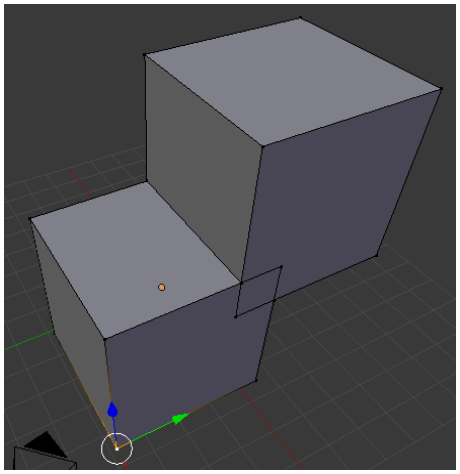


Fig 4.4 - Unione dei due cubi

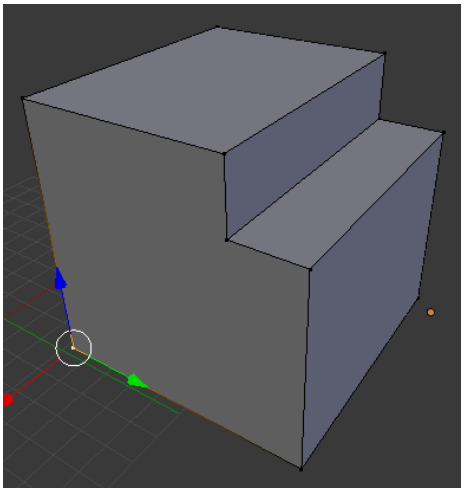


Fig 4.5 - Una delle due differenze tra i due cubi

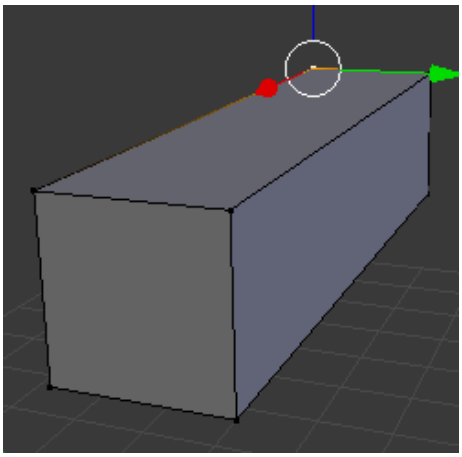


Fig 4.6 -Intersezione dei due cubi

Test 2 (cubo1 cubo3)

La situazione che si presenta in questo test è simile alla precedente, ma vi è una sola zona di tangenza e una porzione del profilo che ricade nel caso di normale intersezione (fig. 4.7).

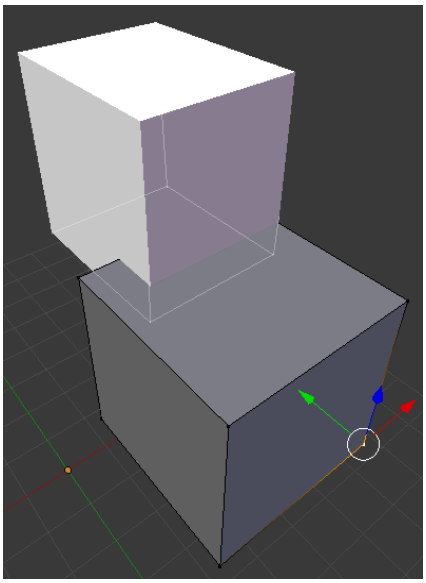


Fig. 4.7 - I due cubi generano una sola zona di tangenza

In figura 4.8 è possibile osservare il profilo di intersezione generato dai due cubi, e come la gestione della faccia tangente venga gestita correttamente anche in presenza di un profilo che non collega sole facce tangenti.

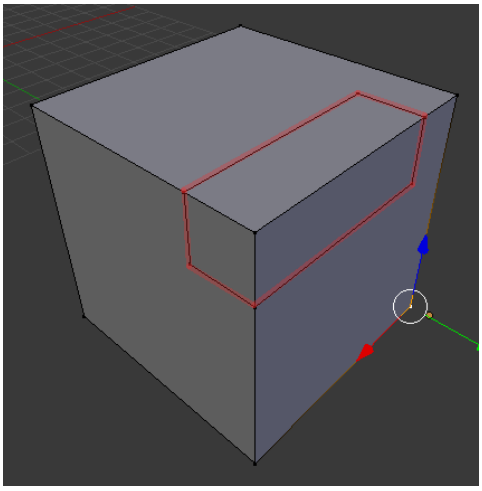


Fig. 4.8 - Profilo di intersezione

Nelle due figure seguenti viene mostrato il risultato di unione e una delle due differenze. Anche in questo caso la gestione è corretta.

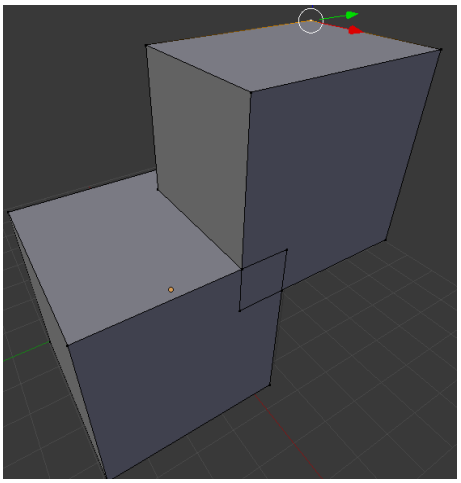


Fig. 4.9 - Unione

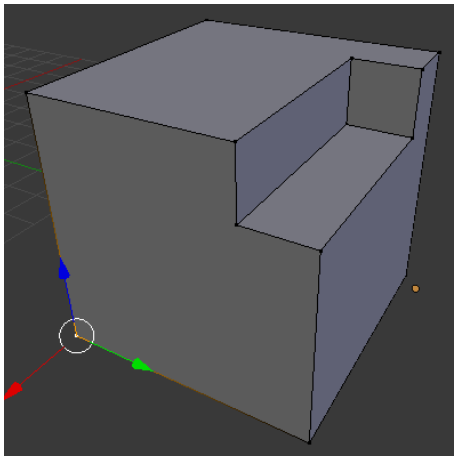


Fig. 4.10 - Differenza

Test 3 (cubo1 cubo4)

Nei due test precedenti, ciascuna regione di tangenza, all'interno della rispettiva faccia del cubo, è individuata da un singolo attraversamento della corrispondente faccia nell'altro cubo (fig. 4.11). In questo caso, la faccia originale viene scomposta in una faccia tangente e in una faccia residua.

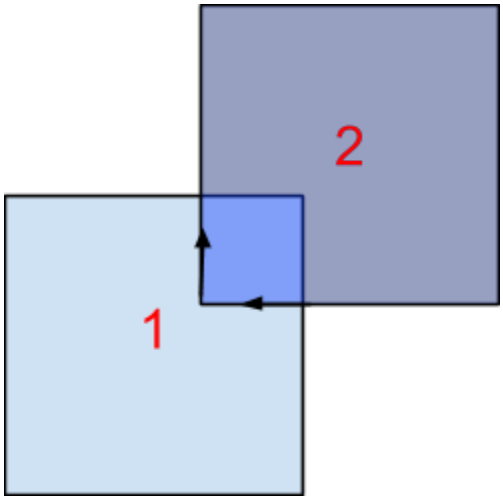


Fig 4.11 - La faccia 2 attraversa la faccia 1 una sola volta, e viceversa

Per verificare la correttezza della soluzione nel caso di attraversamenti multipli di una faccia nell'altra, si è ideato lo scenario di test mostrato in figura 4.12.

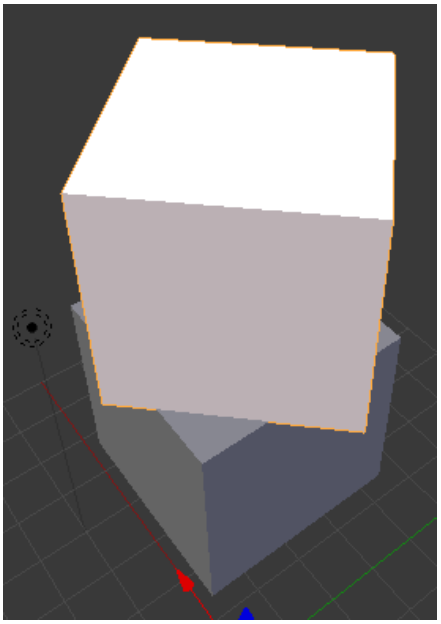


Fig. 4.12 - Le facce dei cubi poste a contatto si attraversano più volte

In questa situazione le due facce si attraversano reciprocamente quattro volte. La fase di inserimento delle facce tangenti produce, per ciascuna delle due facce, una faccia tangente e quattro facce residue (fig 4.13).

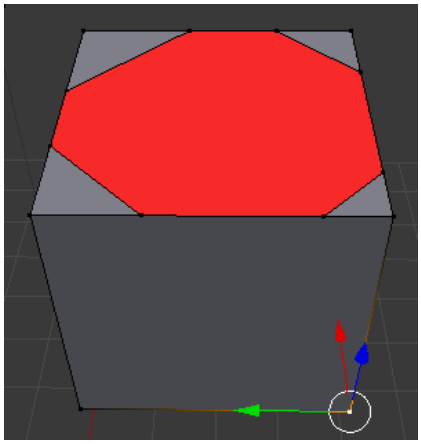


Fig. 4.13 - Il risultato della fase di inserimento delle facce tangenti

In questo caso, non vi è un profilo di intersezione e il risultato è immediato:

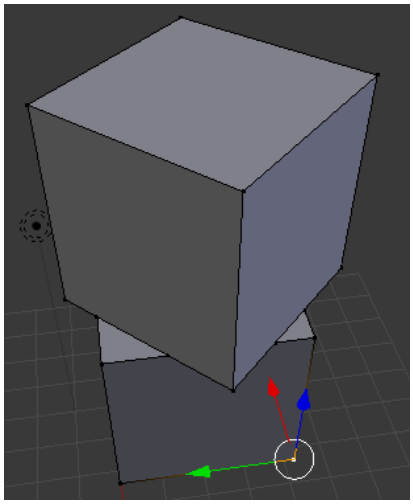


Fig. 4.14 - Unione

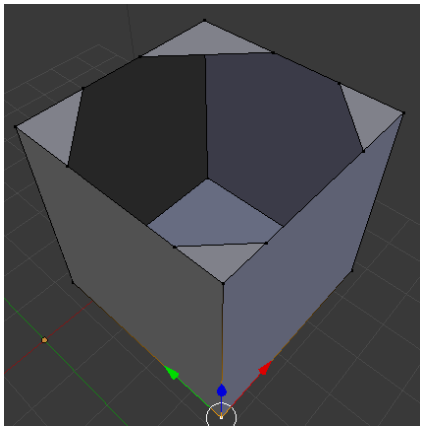


Fig. 4.15 - Una delle due differenze

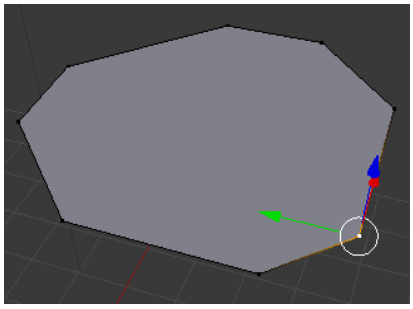


Fig. 4.16 - Intersezione

Test 4 (cubo3 cubo5)

In questo test si mostra cosa accade quando una faccia di una delle due mesh è tangente a più di una faccia dell'altra mesh.
La situazione è simile a quella presentata nel test 2, ma stavolta la sovrapposizione delle facce in tangenza è di tipo uno a molti.

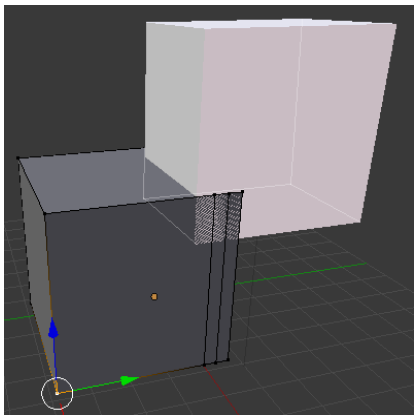


Fig. 4.17 - Tangenza di tipo uno a molti

La gestione delle facce in tangenza, come detto precedentemente, avviene tra coppie di facce e quindi questa situazione non aggiunge alcuna complessità nella procedura di inserimento delle nuove facce. Il risultato di questa fase è mostrato in figura 4.18.

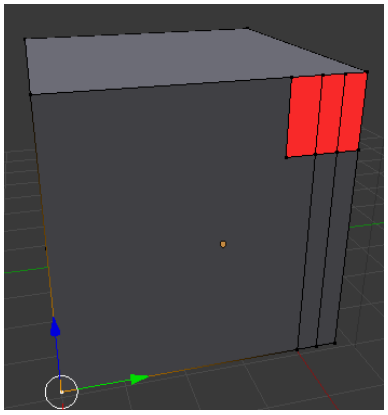


Fig. 4.18 - Facce tangenti inserite nella mesh

Fatta eccezione per la zona di tangenza, il profilo di intersezione è identico a quello visto nel caso del test 2:

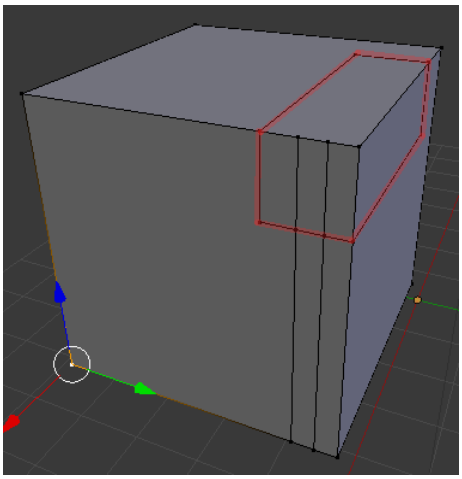


Fig. 4.19 Profilo di intersezione inserito

Infine, si può osservare la coerenza delle mesh in output con quanto è stato appena mostrato:

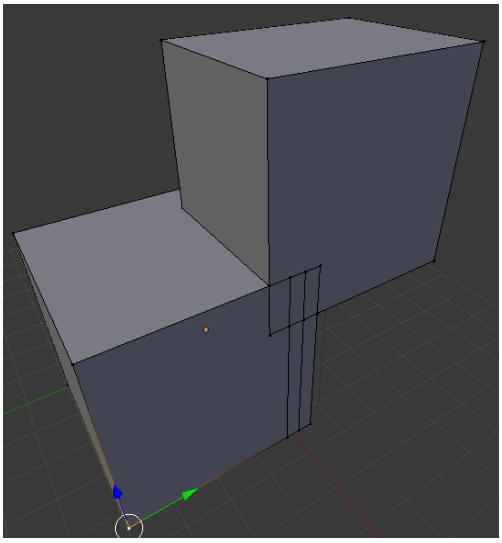


Fig. 4.20 - Unione

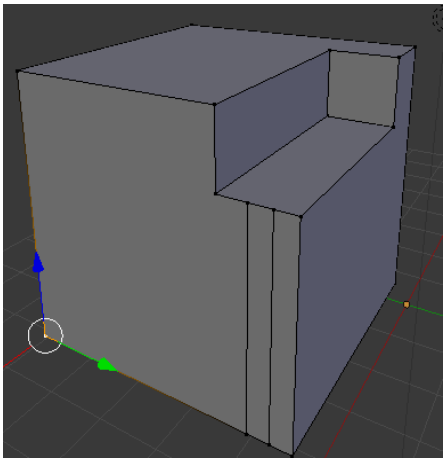


Fig. 4.21 - Una delle due differenze

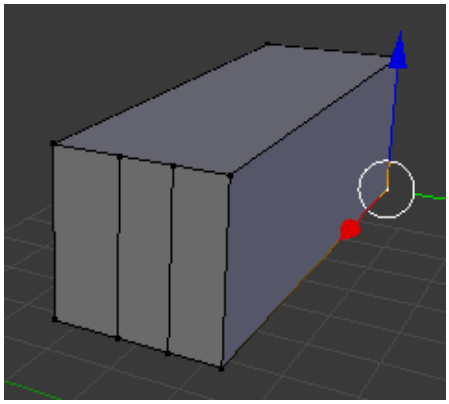


Fig. 4.22 - Intersezione

Test 5 (cubo1 cubo6)

In questo scenario di test si verifica che una zona di tangenza interamente contenuta in una faccia venga correttamente gestita. Le facce in tangenza delle due mesh non presentano dunque attraversamenti reciproci.

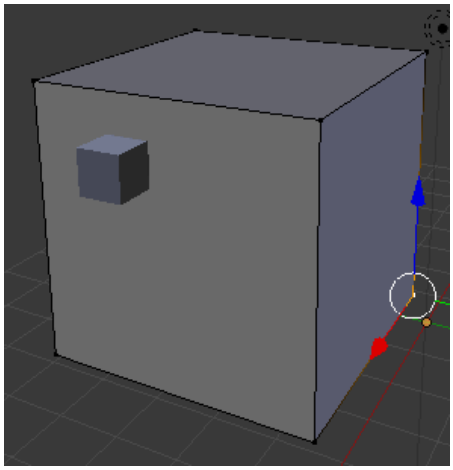


Fig. 4.23 - Una delle due mesh è in tangenza in un faccia dell'altra mesh

L’inserimento della faccia tangente provoca la suddivisione della faccia del cubo più grande in tre nuove facce (fig. 4.24).

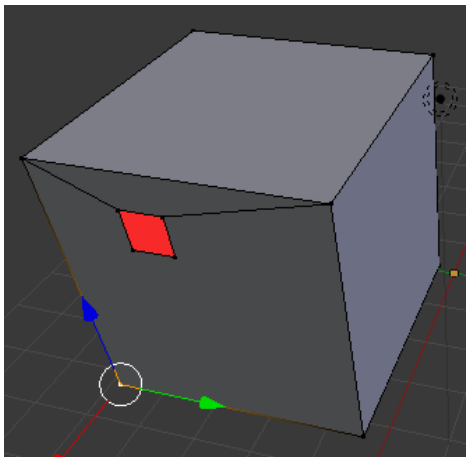


Fig. 4.24 - Inserimento della faccia tangente

Anche in questo caso non vi è un profilo di intersezione. I risultati sono immediati:

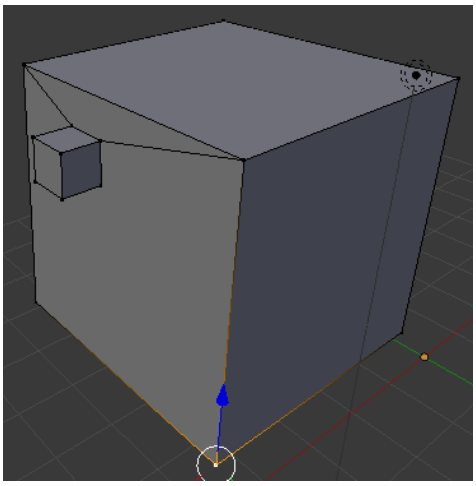


Fig. 4.25 - Unione

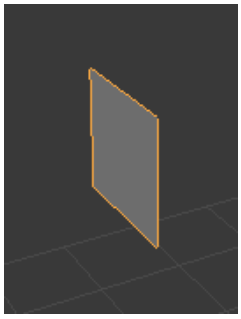


Fig. 4.26 - Intersezione

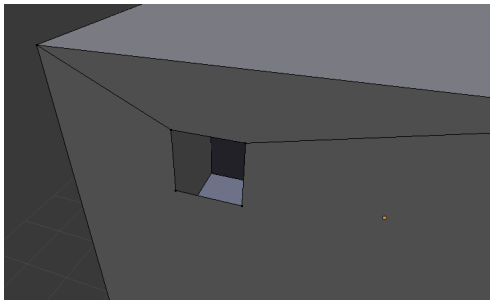


Fig. 4.27 - Una delle due differenze

Test 6 (cubo1 cubo7)

In questo test si illustra l'ultima delle situazioni attualmente gestite da Mesh Craft grazie a questo lavoro: la sovrapposizione totale di due facce (fig. 4.28).

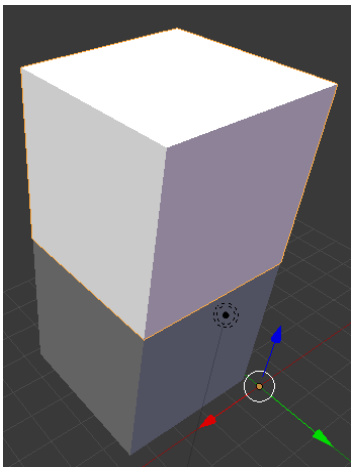


Fig. 4.28 - Tangenza con sovrapposizione totale delle facce

Si è già spiegato nelle precedenti sezioni, che questo è il caso più semplice da trattare. In effetti, non si fa nulla se non marcare le facce come tangenti.

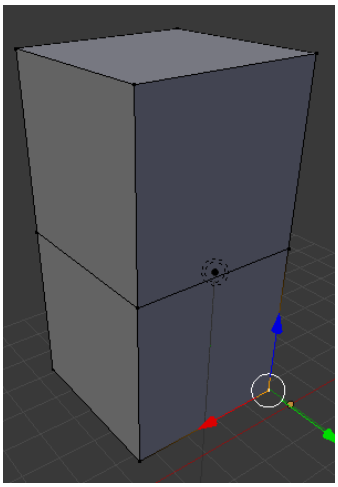


Fig. 4.29 - Unione

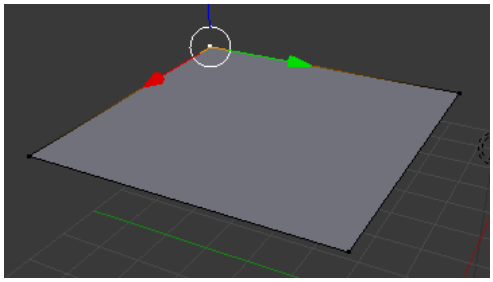


Fig. 4.30 - Intersezione

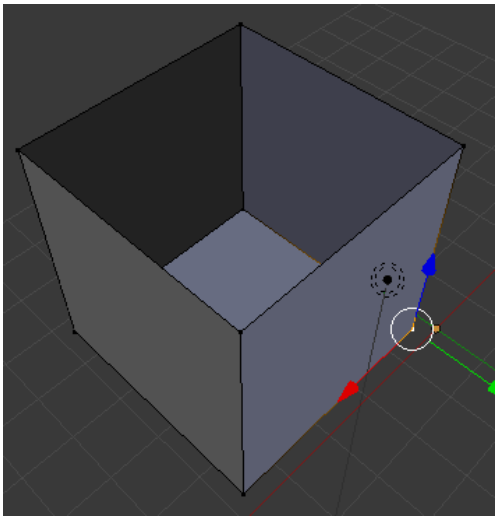


Fig. 4.31 - Una delle due differenze

Test 7 (ala1 corpo)

Questo test si colloca in uno scenario in cui si vuole assemblare un aereo a partire dalle sue componenti. Nello specifico, si vuole unire la metà anteriore del corpo dell'aereo con l'ala sinistra (fig 4.32 e fig. 4.33).

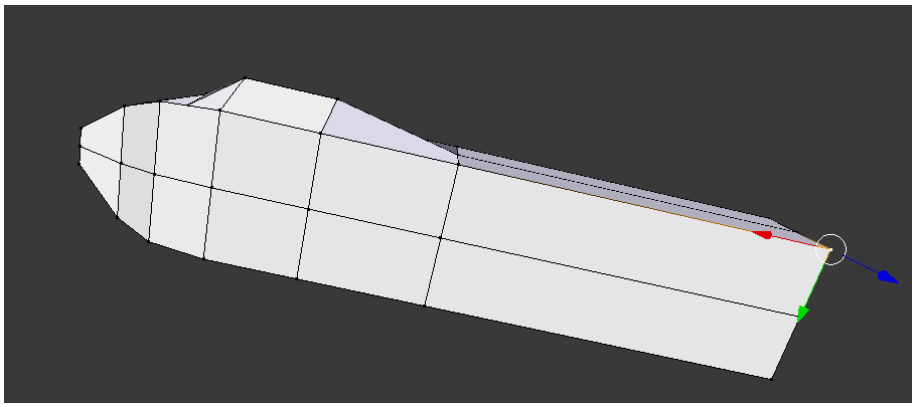


Fig. 4.32 - corpo

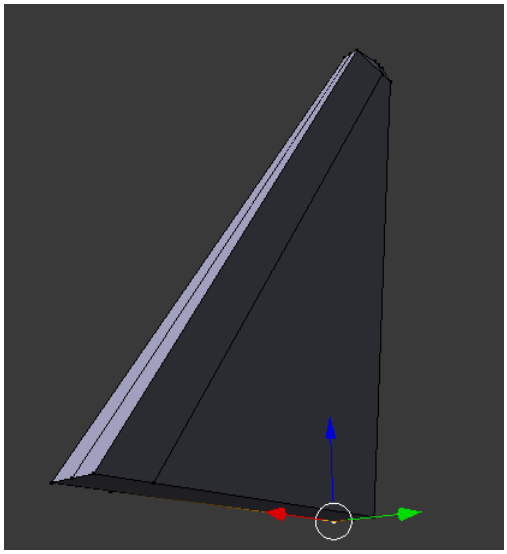


Fig. 4.33 - ala1

Il caso in cui si ricade è quello di una sovrapposizione di tipo uno a molti con una faccia dell'ala in tangenza con quattro facce del corpo dell'aereo.
L'inserimento delle nuove facce tangenti nel corpo dell'aereo genera la mesh mostrata in figura 4.34. La faccia dell'ala subisce la stessa suddivisione individuata dalla zona tangente sulla mesh del corpo.

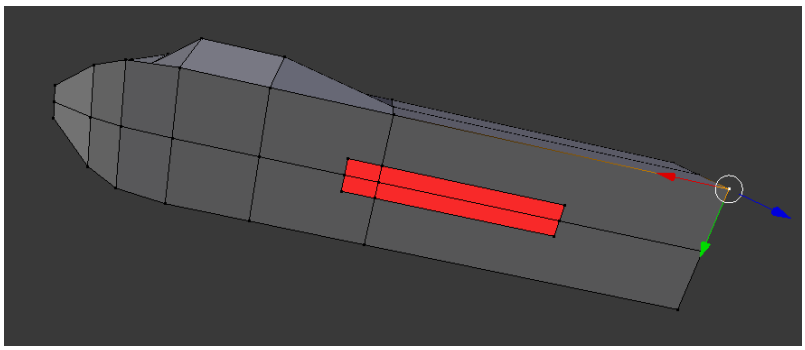


Fig. 4.34 - La zona di tangenza generata nella fase preliminare

L'unione delle due mesh produce il risultato desiderato:

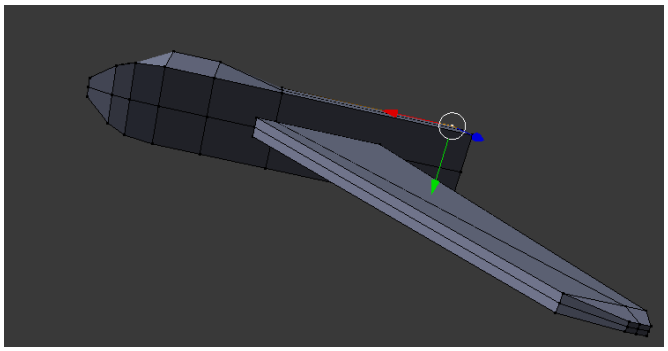


Fig. 4.35 - L'unione delle due mesh

Test 8 (calice2 book)

In questo test si utilizza una mesh con un numero di facce superiore a quello delle mesh utilizzate fino ad ora. Tale mesh rappresenta un calice (fig. 4.36), e in questo scenario viene “poggiato” su un’altra mesh (fig, 4.37) in modo da avere una zona di tangenza tutta interna ad una faccia.

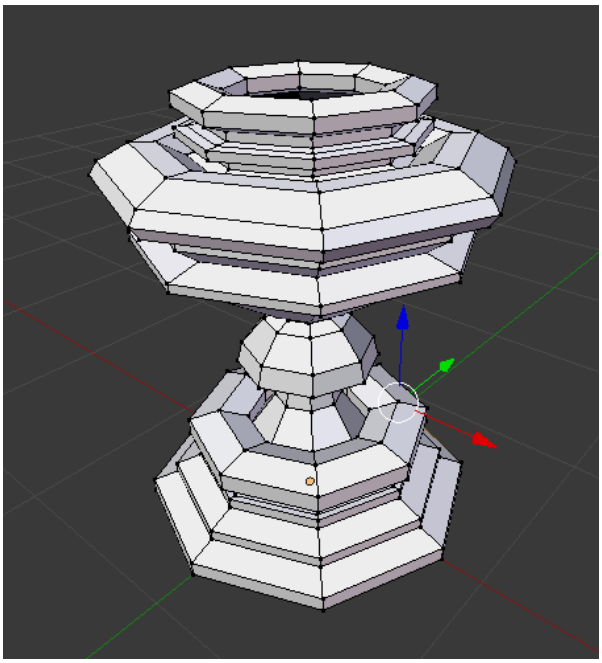


Fig. 4.36 - Il calice

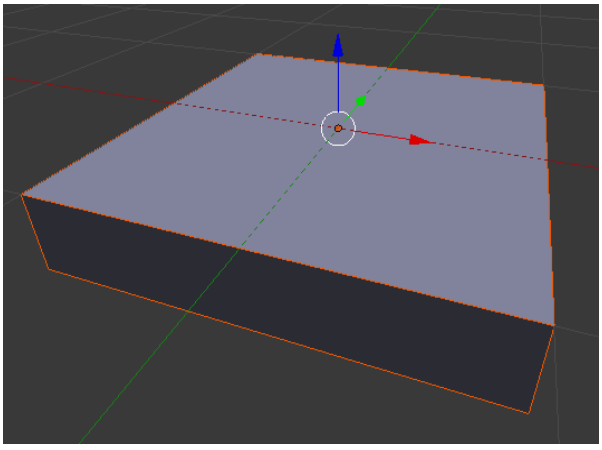


Fig.4.37 - Il libro

La gestione della tangenza avviene, come per gli scenari simili illustrati precedentemente, suddividendo la faccia che contiene la zona di tangenza in tre parti. In figura 4.38 il risultato:

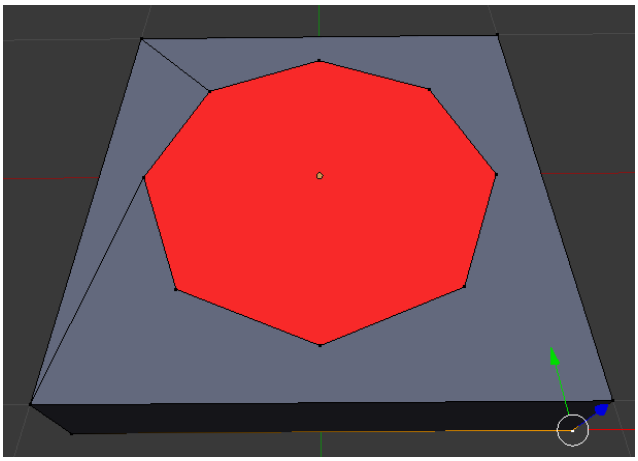


Fig. 4.38 - Inserimento della zona di tangenza

In assenza di un profilo di intersezione, il risultato è immediato:

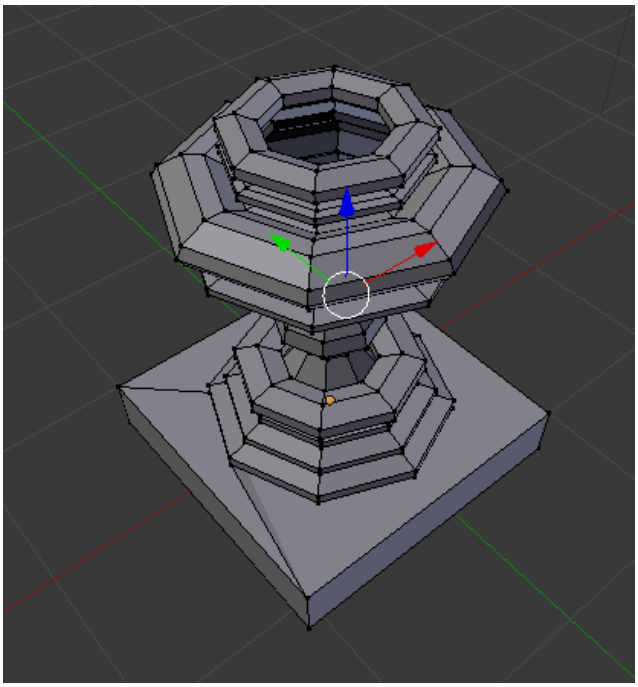


Fig. 4.39 - Unione delle due mesh

Test 9 (calice2 box)

Simile al test precedente, questo scenario mostra una situazione di tangenza in cui una delle due mesh si trova all'interno dell'altra mesh.
La gestione della tangenza è identica a quella del test 8 ma, durante la procedura di classificazione, il calice viene questa volta individuato come appartenente al volume di intersezione e i risultati sono differenti. In figura 4.40 la differenza tra il box e il calice:

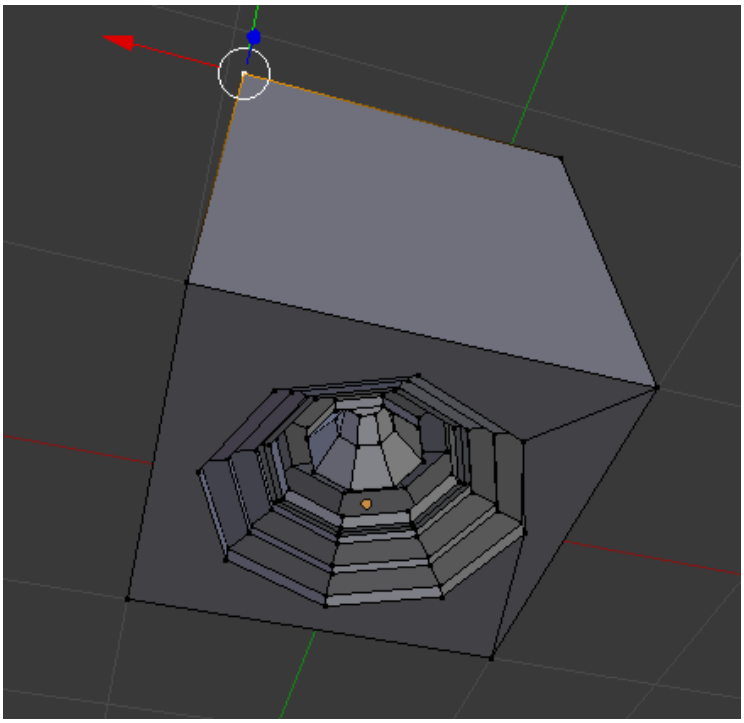


Fig. 4.40 - Differenza tra il box e il calice

In questa sezione sono stati illustrati diversi casi di test, evidenziandone la corretta gestione da parte di Mesh Craft.
I test effettuati hanno utilizzato prevalentemente mesh semplici dato che, ai fini della gestione della tangenza, ciò che ha rilevanza sono le caratteristiche di sovrapposizione delle facce tangenti.
Come sottolineato all'inizio di questa sezione, la fase di test non si considera conclusa con la trattazione degli scenari che sono stati presentati, poichè altre situazioni, non considerate in sede di analisi, possono emergere in fase di test.

4.2 Confronto con Blender

Al fine di avere un termine di confronto per i risultati, sono stati eseguiti gli stessi test in Blender (versione 2.70), il quale gestisce le operazioni Booleane grazie alla libreria Carve.
In generale, Blender calcola correttamente il risultato di unione per tutti i test con i cubi e include le facce tangenti nei volumi di intersezione. Tuttavia, vi sono comportamenti diversi, tra Blender e Mesh Craft, nel calcolo dell'intersezione e delle differenze.
Innanzitutto, Blender non tratta le facce tangenti come facenti parte dell'intersezione se non legate ad un volume: due mesh, come quelle del test 3, non producono nessun risultato di intersezione. Inoltre, le caratteristiche del risultato sono legate all'ordine delle mesh: in Blender occorre prima selezionare una mesh e poi procedere con un operazione Booleana selezionando una seconda mesh, l'ordine di selezione delle mesh è importante dato che può produrre risultati diversi.
Quanto detto viene illustrato di seguito con maggior dettaglio.

Se si ripete il test 5 con Blender, si può osservare che la faccia tangente viene riconosciuta e inserita (fig. 4.41) in modo simile a quanto si ottiene con Mesh Craft. Tuttavia, se si tenta di calcolare la differenza del cubo piccolo dal cubo grande, ciò che si ottiene è quello che si vede in figura 4.41: viene modificata la struttura di una faccia ma non ne viene rimossa alcuna parte. Il risultato corretto che si sarebbe dovuto ottenere, calcolato da Mesh Craft, è mostrato in fig. 4.27.

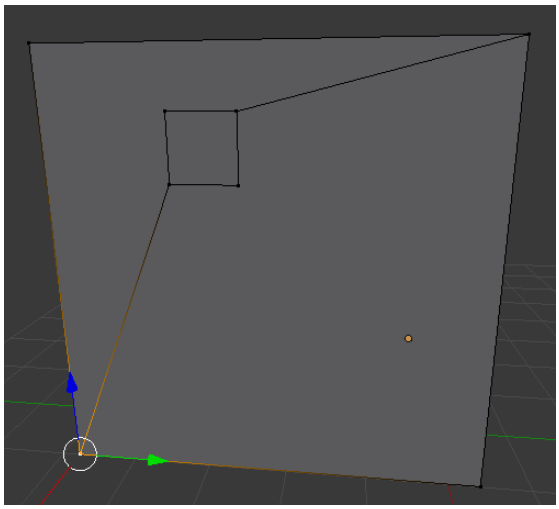


Fig. 4.41 - Intersezione per i cubi del test 5 in Blender

Al fine di sintetizzare il confronto per i test con i cubi si riportano i risultati ottenuti per Blender e Mesh Craft in tabella.

Blender	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
A ∪ B	OK	OK	OK	OK	OK	OK
A ∩ B	OK	OK	KO	OK	KO	KO
A - B	OK	OK	KO	OK	KO	KO
B - A	OK	OK	KO	OK	KO	KO
Mesh Craft	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6
A ∪ B	OK	OK	OK	OK	OK	OK
A ∩ B	OK	OK	OK	OK	OK	OK
A - B	OK	OK	OK	OK	OK	OK
B - A	OK	OK	OK	OK	OK	OK

Per quanto riguarda il test dell'aereo, vi sono dei comportamenti differenti. In questo caso il calcolo dell'unione fallisce, mentre il risultato di intersezione presenta due possibilità: se si effettua l'intersezione del corpo con l'ala, si ottiene il risultato mostrato in figura 4.42; se invece si effettua l'intersezione dell'ala con il corpo, si ottiene correttamente la sola faccia di intersezione (fig. 4.43). Si può notare come la topologia della zona di tangenza sia diversa nelle due intersezioni, e che questa appare comunque più complessa di quella costruita in Mesh Craft.

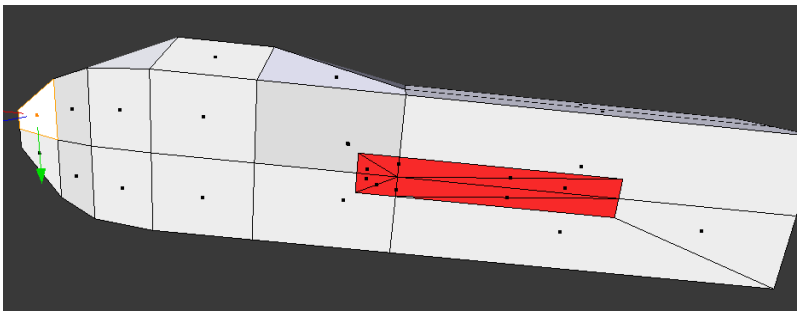


Fig. 4.42 - Intersezione del corpo con l'ala

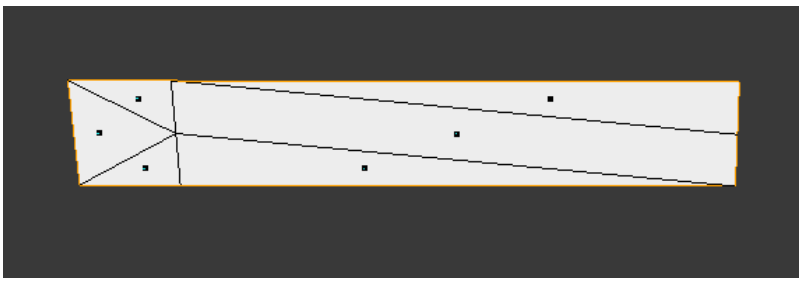


Fig. 4.43 - Intersezione dell'ala con il corpo

In sintesi:

	Blender	Mesh Craft
A ∪ B	KO	OK
A ∩ B	OK parziale	OK
A - B	KO	OK
B - A	KO	OK

Passando ad un ultimo test di confronto, il caso del calice posto sopra il libro, Blender non riesce a determinare correttamente nessun risultato. In figura 4.44 si può vedere come nel risultato di unione manchi la parte inferiore del calice e che questa sia assente anche nel calcolo della differenza (fig. 4.45).

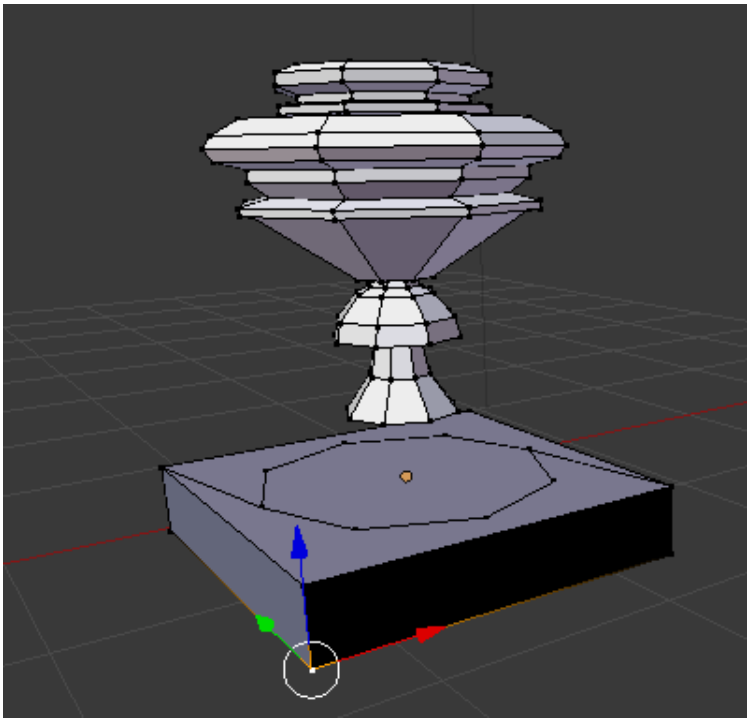


Fig. 4.44 - Unione del calice con il libro calcolata da Blender

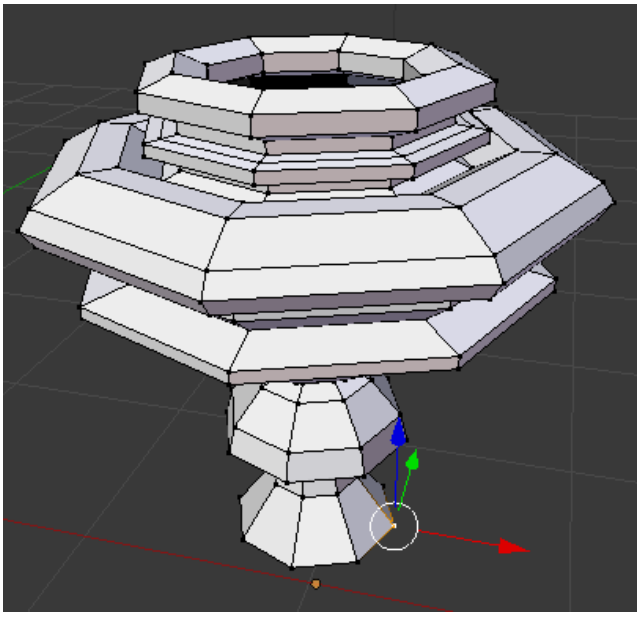


Fig. 4.45 - Differenza tra il calice e il libro calcolata da Blender

La parte inferiore è invece presente nel risultato di intersezione (fig. 4.46). Questo fa pensare, coerentemente con quanto ottenuto per unione e differenza, che questa porzione del calice venga erroneamente classificata come appartenente all'intersezione delle due mesh e che per questo motivo non sia presente nei due risultati precedenti.

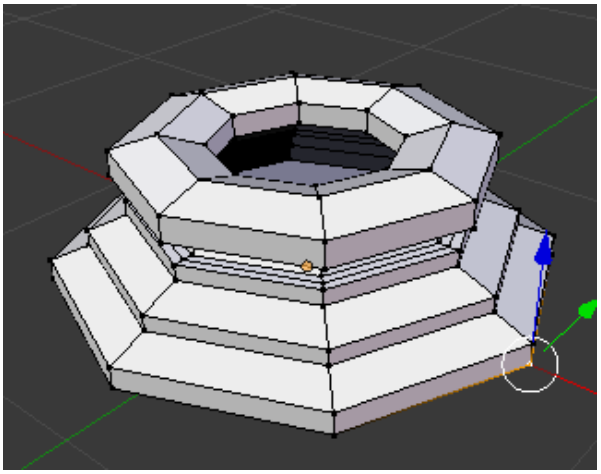


Fig. 4.46 - Intersezione tra il calice e il libro calcolata da Blender

Una precisazione che occorre fare su questo test, è che la mesh del calice non è costituita da un unico pezzo ma da più pezzi (fig. 4.47). Se ne deduce quindi che Blender ha dei problemi con la gestione di questo tipo di mesh, a differenza di Mesh Craft.

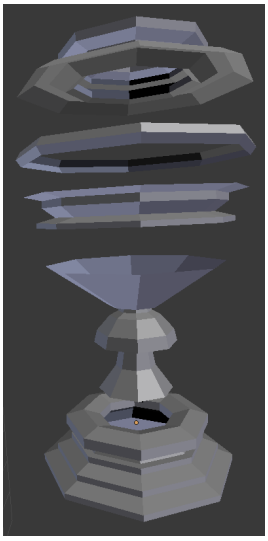


Fig. 4.47 - I pezzi che formano il calice

In sintesi:

	Blender	Mesh Craft
A ∪ B	KO	OK
A ∩ B	KO	OK
A - B	KO	OK
B - A	KO	OK

4.3 Intersezioni multiple

Per completezza di trattazione, si riportano in questa sezione due esempi di intersezioni multiple. Nel primo esempio, si effettua l'intersezione di un cilindro (fig. 4.48) con un oggetto che può essere descritto come un cilindro con un gancio (fig. 4.49).

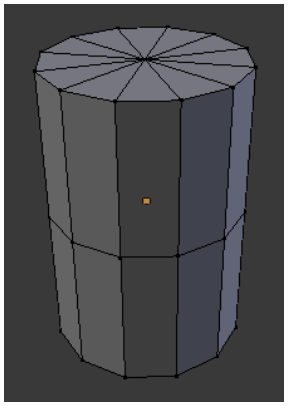


Fig. 4.48 - Il cilindro

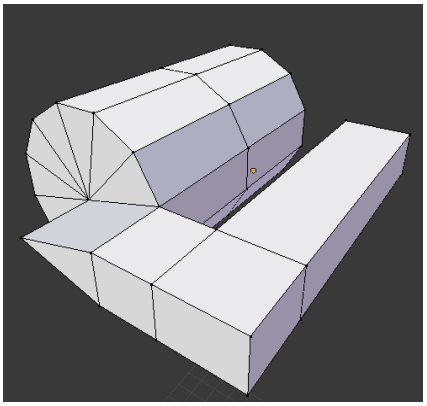


Fig. 4.49 - Il "cilindro con gancio"

Il cilindro viene collocato nei pressi del gancio per produrre i seguenti risultati:

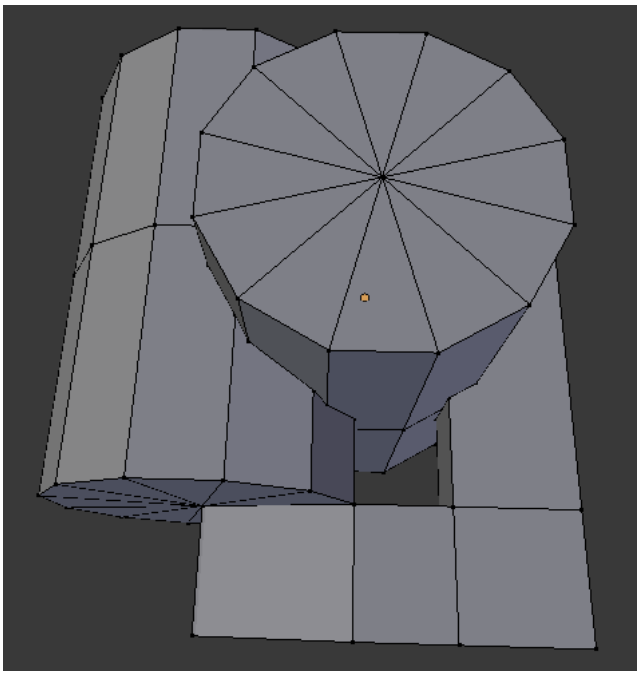


Fig. 4.50 - Unione

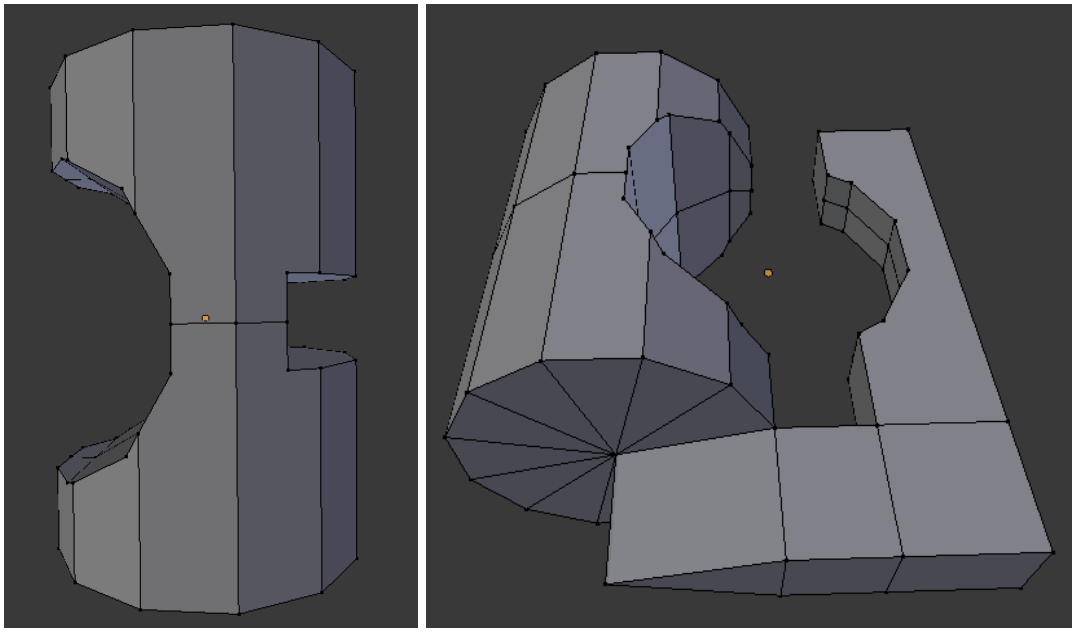


Fig. 4.51 - Le due differenze

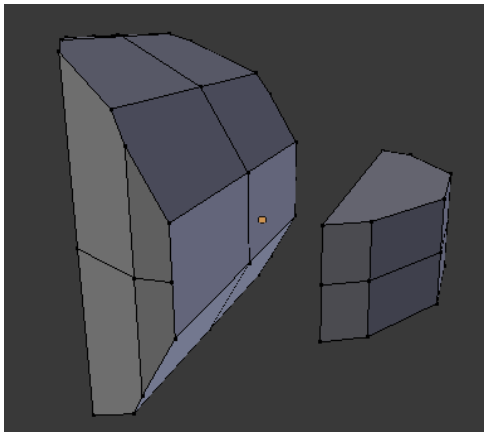


Fig. 4.52 - Intersezione

Per quanto riguarda il secondo esempio, si inserisce una sfera in un cubo, di modo che la prima intersechi ciascuna delle sei facce del cubo. Nelle figure seguenti i risultati:

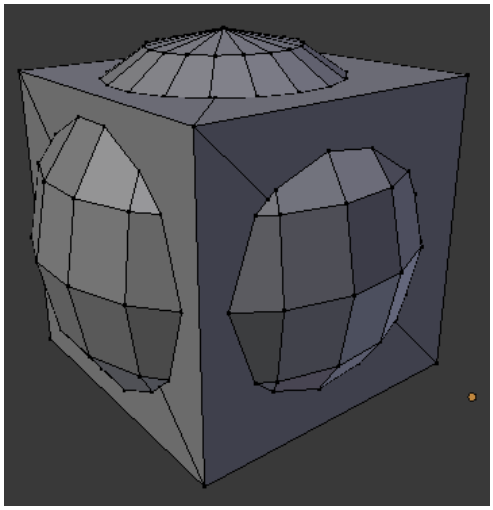


Fig. 4.53 - Unione

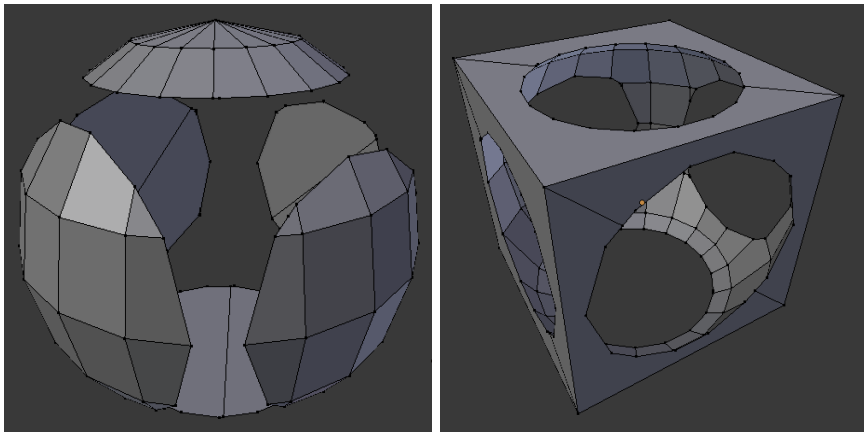


Fig. 4.54 - Le due differenze

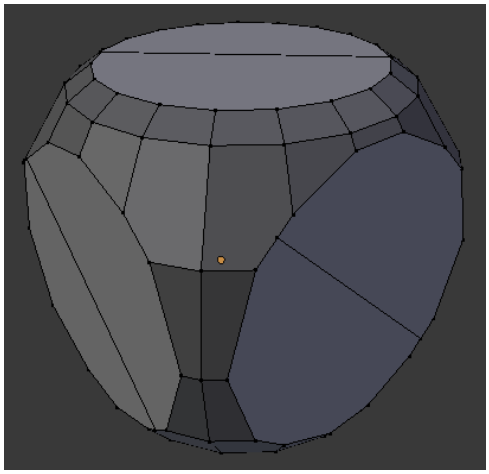


Fig. 4.55 - Intersezione

4.4 Operazioni Booleane con Mesh Craft

Al fine di concludere questa parte relativa alla sperimentazione, e per consentire di dare uno sguardo all'interfaccia di Mesh Craft, si riportano due esempi di operazioni Booleane.

Nel primo esempio, si vuole creare una mesh che rappresenta un salvadanaio a partire dalla mesh di un maialino. Per fare questo, si inserisce sul dorso di quest'ultimo un box e si invoca la procedura di calcolo delle operazioni Booleane (fig. 4.56).

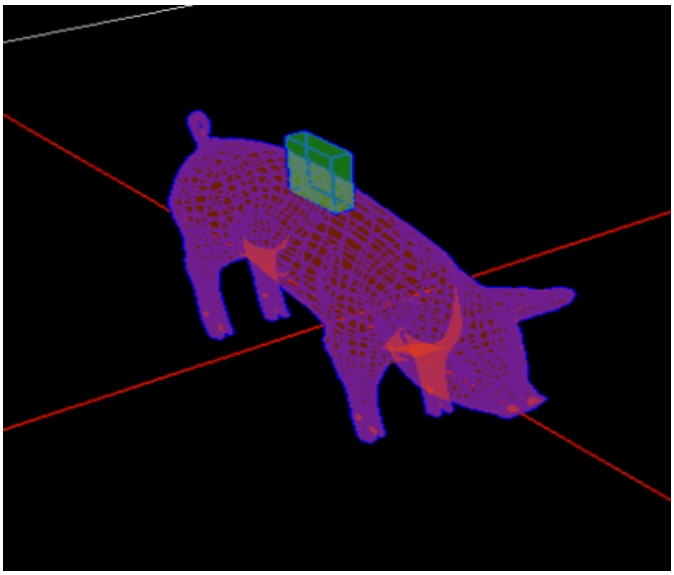


Fig. 4.56 - Si inserisce il box sul dorso del maialino

In figura 4.57 si può vedere come si presenta Mesh Craft: il menu testuale riporta alcune scorciatoie da tastiera, ma tutta l'interazione con gli oggetti nello spazio di lavoro avviene mediante il Leap Motion Controller.

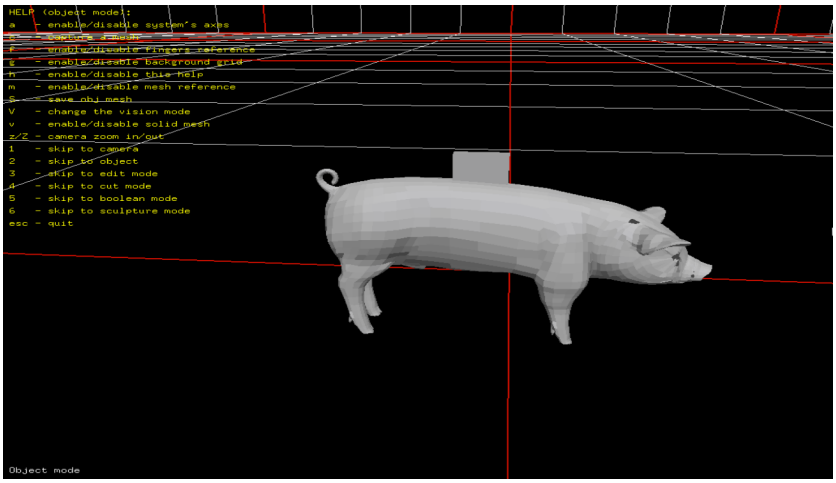


Fig. 4.57 - Le due mesh in Mesh Craft

L'oggetto che ci si è proposti di creare si ottiene dalla differenza delle due mesh e il risultato è visibile in figura 4.58.

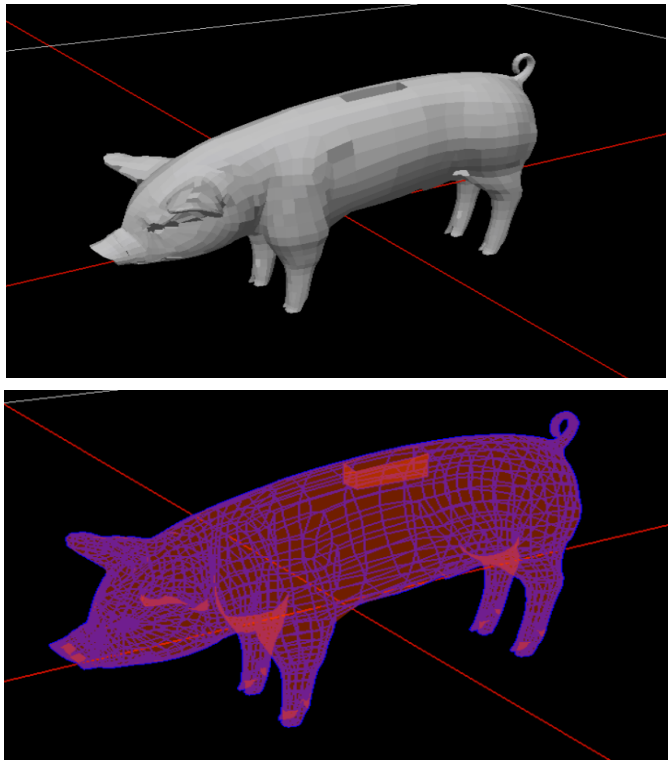


Fig. 4.58 - Il salvadanaio

Il secondo esempio ha come scopo quello quello di modificare il coperchio di una teiera (fig. 4.59). Per fare questo, si combina la teiera con una sfera.

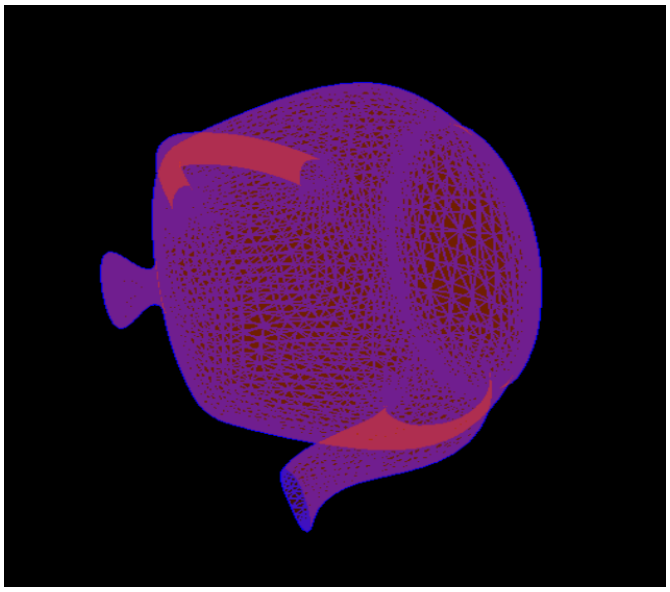


Fig. 4.59 - La teiera

Collocando la sfera sul coperchio di modo da inglobare il pomo originale della teiera, è sufficiente effettuare l'unione dei due oggetti per ottenere il risultato desiderato.

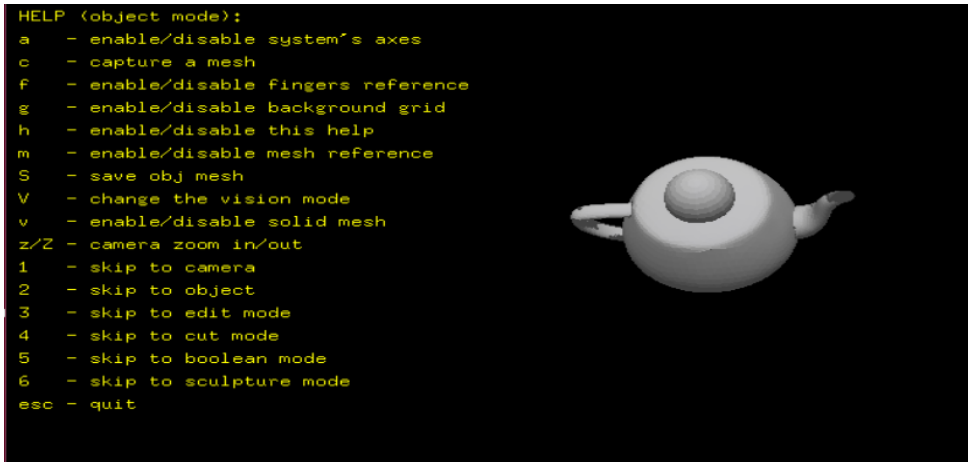


Fig. 4.60 - L'unione della sfera con la teiera e il dettaglio del menu di Mesh Craft

4.5 Casi non gestiti

Vi sono due situazioni che, nonostante fossero state previste durante la fase iniziale di questo lavoro, non vengono attualmente gestite.

Il primo scenario è già stato accennato in precedenza: la tangenza di una faccia totalmente interna a una faccia dell'altra mesh con un punto di contatto (fig. 4.61).

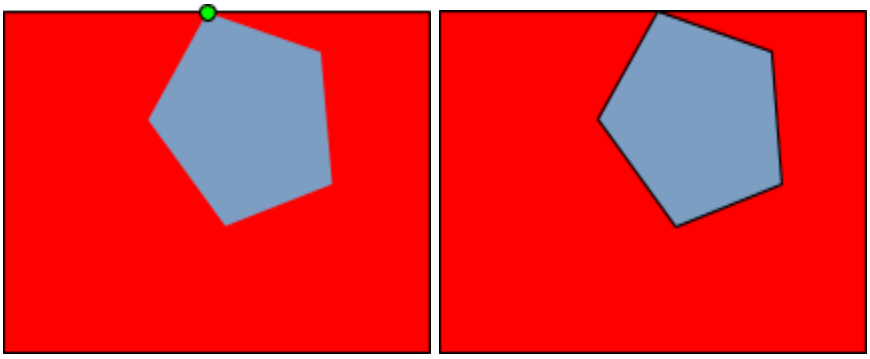


Fig. 4.61 - La sovrapposizione di una faccia, interamente contenuta nell'altra ma con un punto di contatto, dovrebbe causare la suddivisione della faccia esterna in due facce.

La gestione che si dovrebbe avere è quella riportata in figura 4.61 ma al momento non è contemplata nella logica di gestione.

Il secondo scenario è invece un caso di sovrapposizione parziale in cui i punti che individuano la suddivisione di una faccia sono vertici dell'altra faccia (fig. 4.62).

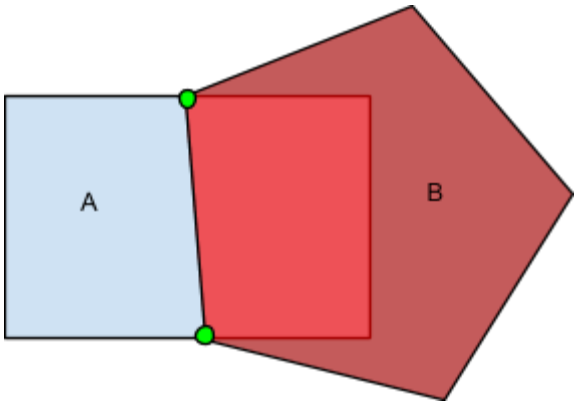


Fig. 4.62 - La faccia A non viene suddivisa correttamente perchè i punti che individuano la linea di taglio sono vertici della faccia B.

Questa situazione non viene gestita perchè la procedura al momento non considera correttamente i vertici delle facce nel caso in cui questi siano anche punti di intersezione tra i due poligoni.

4.6 Tempi di esecuzione

Per concludere questa sezione, si vuole fornire un'idea dei tempi di elaborazione necessari per eseguire le operazioni Booleane con Mesh Craft. In tabella vengono riportati, tra tutti quelli mostrati in questo capitolo, i test con mesh più dettagliate e i loro rispettivi tempi di esecuzione. Tutti i test sono stati eseguiti su una macchina con frequenza del processore pari a 2 GHz e 2 GB di RAM.

Test	Facce Mesh 1	Facce Mesh 2	T
calice, book	377	6	0,26 s
maiale, box	3555	6	2,85 s
teiera, sfera	8480	480	6,72 s

Per fornire un maggior livello di dettaglio sui tempi di esecuzione, si mostrano anche i tempi relativi alla funzione che calcola le informazioni addizionali della mesh (*Mod_LoadSubdivMesh*) a partire dalla lista dei suoi vertici e delle sue facce.

Vertici	Lati	Facce	Tempo
13	20	9	0,12 ms
32	108	72	0,48ms
424	808	384	2,95ms
962	1920	960	5,56ms

Naturalmente, il tempo di esecuzione della funzione *Mod_LoadSubdivMesh* è incluso nel tempo di esecuzione totale.

Conclusioni

L'obiettivo di questo lavoro è stato quello di introdurre nel progetto Mesh Craft ciò che era stato fatto con il progetto Mesh Glue, e di estendere la logica delle operazioni Booleane per consentire la gestione delle zone di tangenza.
Riguardo all'introduzione delle funzionalità di Mesh Glue nel nuovo progetto, ci si può ritenere soddisfatti, anche se è sicuramente necessaria l'introduzione di una logica di selezione del risultato da ottenere (tra i quattro possibili risultati) nel menu di Mesh Craft.
Per quanto riguarda invece la gestione della tangenza nelle operazioni Booleane, si sono ottenuti dei risultati validi e robusti, e l'introduzione di nuove facce legate alla tangenza avviene con una procedura che minimizza le modifiche sulle facce preesistenti, evitando di complicare eccessivamente la topologia della Mesh nelle zone tangenti (come invece può accadere con Blender nel test dell'aereo). Tuttavia, come è stato detto nel capitolo sulla sperimentazione, la fase di test non si può considerare conclusa con questo lavoro, poichè potrebbero esistere degli scenari che non sono stati valutati in sede di analisi. Per quanto riguarda i due scenari non gestiti presentati alla fine del capitolo precedente, la loro corretta gestione comporta solo lievi modifiche alla procedura di gestione delle facce tangenti. In particolare, per lo scenario di faccia interna con singolo punto di intersezione, è sufficiente adattare la gestione del caso evidenziato nel test 5 in modo da utilizzare il punto di intersezione nella costruzione delle nuove facce, mentre, per quanto riguarda l'altro scenario, è sufficiente modificare la procedura di costruzione delle sequenze di punti in modo che prenda anche i vertici delle facce se questi coincidono con i punti di intersezione.
Ulteriori aspetti da tenere in conto sono quelli legati alle performance. Questi aspetti discendono direttamente da Mesh Glue. Innanzitutto, sebbene i tempi di elaborazione per le informazioni supplementari relative alle mesh su cui si opera (mediante la funzione *Mod_LoadSubdivMesh*) siano molto bassi, queste vengono aggiornate più volte, mediamente dalle 5 alle 10 volte. Inoltre la maggior parte dei controlli effettuati per determinare adiacenze tra elementi di mesh distinte e intersezioni tra oggetti, non sono per ora assolutamente ottimizzati, il che porta a condurre numerose ricerche lineari che contribuiscono ad innalzare il costo computazionale [1].

A fronte di ciò, oltre ad ottimizzare le procedure, si può pensare di modificare la procedura di calcolo delle intersezioni per evitare il confronto di tutte le coppie lato-faccia delle due mesh. Ancora, si potrebbero memorizzare le informazioni relative alle intersezioni direttamente nella struttura dati della mesh, in modo da potersi appoggiare direttamente a questa durante le procedure di ricostruzione del profilo e del suo inserimento. Per quanto riguarda la procedura di gestione della tangenza, questa si appoggia sulle strutture dati delle due mesh e non fa uso di strutture aggiuntive.

Infine, ricordando che lo scopo originale di Mesh Glue era quello di produrre un applicativo in grado di gestire mesh non strutturate al fine di ottenere delle buone mesh a cui applicare schemi di suddivisione, si mostra adesso, per completezza di trattazione, il raffinamento mediante schemi di suddivisione del primo test (fig. 5.1). L'operazione è stata eseguita in Blender.

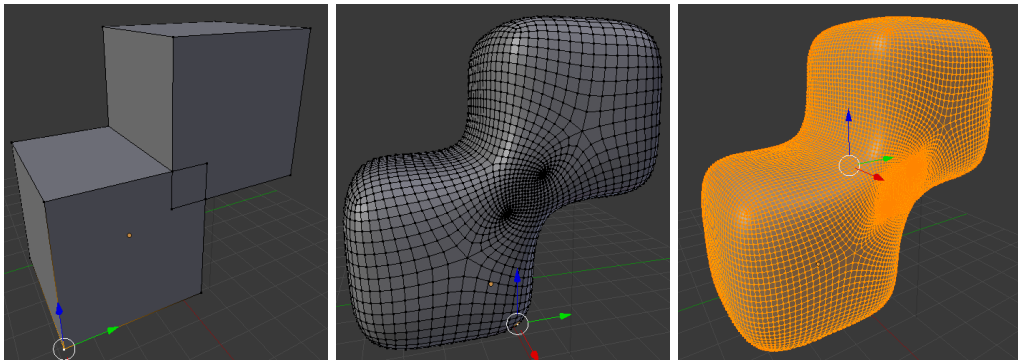


Fig. 5.1 - Raffinamento del risultato di unione ottenuto nel test 1 mediante Suvdivision Surface

Come si è osservato nel lavoro originale [1], la curvatura delle superfici raffinate è decisamente influenzata dalla forma delle facce originali; per ridurre l'insorgenza di vertici straordinari, si può pensare di arricchire le tecniche di unione con una sorta di raffinamento locale delle facce interessate dalla poligonale.

Bibliografia

[1] Modellazione con Mesh non strutturate
Flavio Bertini; 2007, Tesi di laurea in Informatica, Università di Bologna

[2] Fast and accurate evaluation of regularized Boolean operations on triangulated solids
F.R. Feito, C.J. Ogayar, R.J. Segura, M.L. Rivero; Computer-Aided Design
Volume 45, Issue 3, March 2013, Pages 705–716

[3] Fast and robust Booleans on polyhedra
Songgang Xu, John Keyser; Computer-Aided Design Volume 45, Issue 2, February 2013, Pages 529–534

[4] Efficient Boolean Operation on Manifold Mesh Surfaces
Ming Chen, Xiao Yu Chen, Kai Tang and Matthew M. F. Yuen; Computer-Aided Design and Applications Volume 7, Issue 3, 2010, Pages 405-415

[5] A topologically robust algorithm for Boolean operations on polyhedral shapes using approximate arithmetic
J.M. Smith, N.A. Dodgson; Computer-Aided Design Volume 39, Issue 2, February 2007, Pages 149–163

[6] Simple and Robust Boolean Operations for Triangulated Surfaces
Gang Mei and John C. Tipper; 2013, arXiv:1308.4434 [cs.CG]

[7] Reasoning Boolean operation based modeling for heterogeneous objects
W. Sun, X. Hu; Computer-Aided Design Volume 34, Issue 6, May 2002, Pages 481–488

[8] An algorithm reducing 3D Boolean operations to a 2D problem: concepts and results
Yvon Gardan and Estelle Perrin; Computer-Aided Design Volume 28, Issue 4, April 1996, Pages 277–287

[9] Determining if a point is inside a polygon
Paul Bourke; 1987, http://paulbourke.net