# BAB II

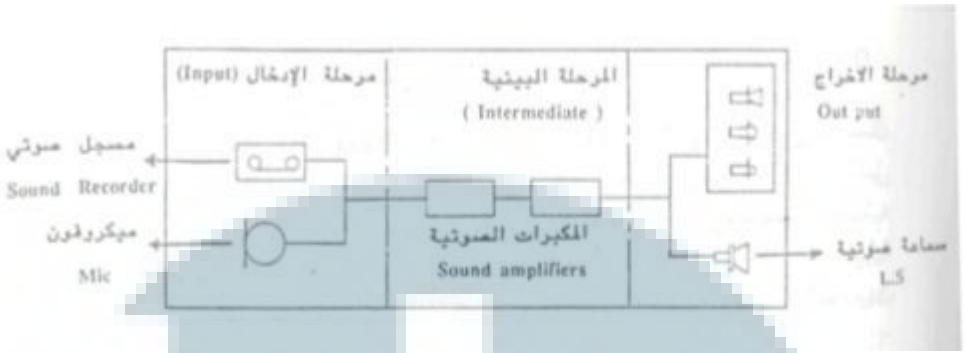
# TINJAUAN PUSTAKA

## 2.1. *Sound System*

Menurut Bob McCarthy, *Sound system* adalah kumpulan dari beberapa perlengkapan yang dihubungkan secara bersama-sama dan digunakan untuk memperkuat sebuah sumber suara dan menyalurkannya kepada sebuah tempat, atau bangunan. Fungsi lainnya yaitu bisa digunakan untuk memperjelas atau menjernihkan sumber suara tersebut.

Banyak pakar audio profesional yang telah mencoba untuk menyalurkan sinyal suara melalui suatu tempat tertentu dalam waktu yang sama mengurangi kualitas dari sumber suara tersebut. Namun banyak masalah yang terjadi saat suara tersebut mulai disiarkan seperti *feedback* saat penyiaran, suara yang tidak stabil dan banyak gangguan yang lain.

Untuk mempermudah, cara kerja *sound system* sebenarnya ialah memilih sumber suara melalui alat-alat tertentu (*input devices*) lalu menuju sebuah perangkat lain (*intermediate devices*) untuk diatur, dikompres dan disaring. Dan kemudian sinyal suara tersebut dipancarkan kepada *speaker* atau pemancar suara.




Gambar 2.1. Tabel Penyaluran Suara
(*Design and Optimization: Modern Techniques and Tools for Sound system Design and Alignment*, hal. 1)

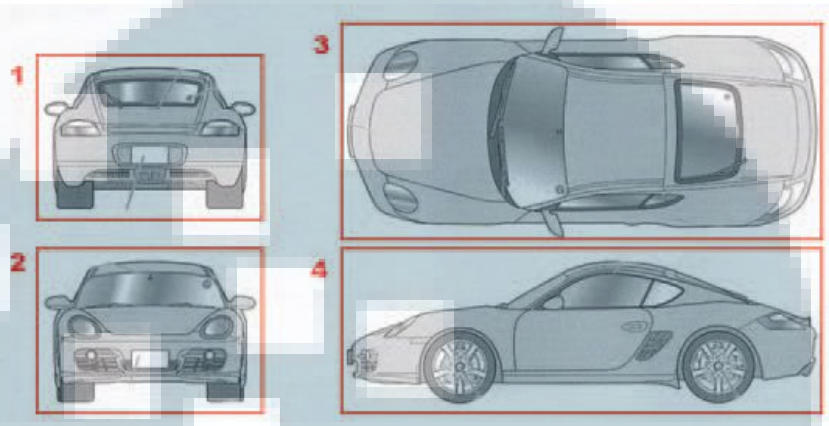
Menurut McCarthy, ada 3 (tiga) model transmisi yaitu:

1. Tahap *Input* yaitu proses perubahan dari suara menjadi elektrik melalui *mic* ataupun alat perekam suara yang lain.
2. Tahap *Intermediate,* sumber suara yang telah diubah menjadi elektrik diperkuat dan disaring kembali untuk mendapatkan suara yang semaksimal mungkin.
3. Tahap *Output*, tahap ini adalah tahap terakhir dimana suara yang telah diproses dari tahap *intermediate*, untuk dapat dikonversi lagi menjadi suara yang bisa didengar kembali.

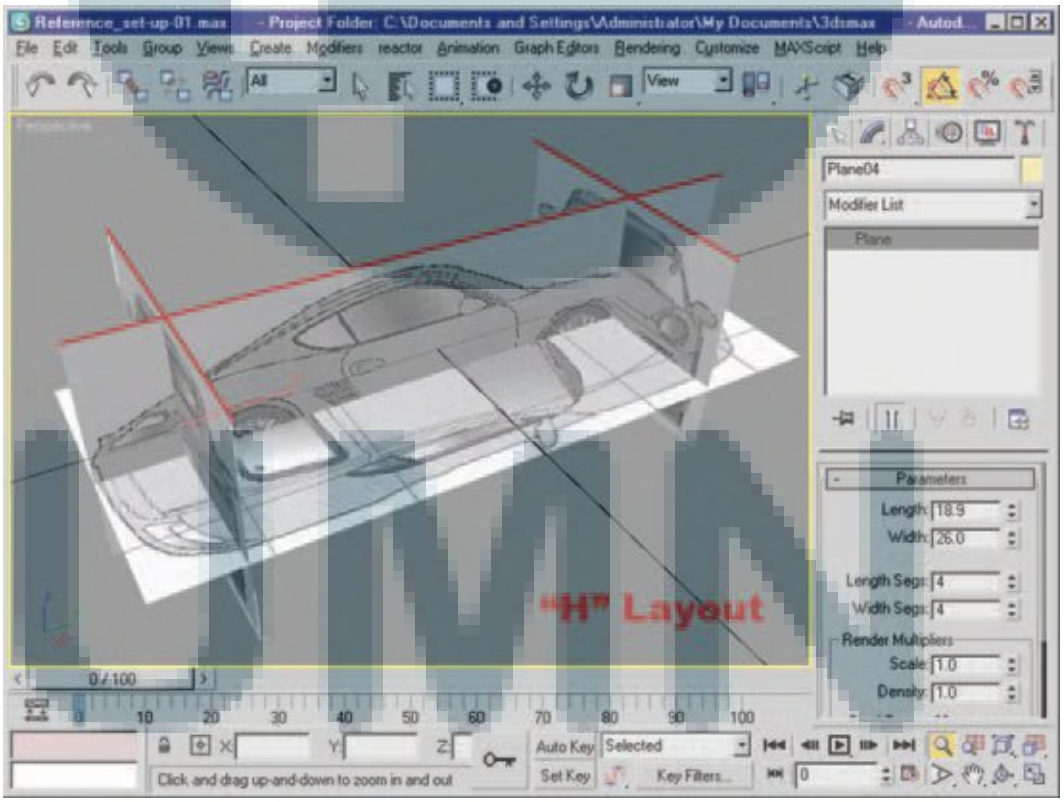
## 2.2. *Image References*

Dikutip dari buku *Poly-Modeling with 3Ds Max Thinking Outside of the Box*, dengan 1 (satu) set referensi yang lengkap dan jelas, serta konsep yang matang, dapat dibuat model yang rumit dengan mudah. Referensi gambar sangat diperlukan bila ingin membuat konsep 3D yang detail dan sesuai dengan benda aslinya. Semakin banyak referensi yang dimiliki, maka semakin mudah juga untuk dapat membuat model yang detail dan berkualitas tinggi.

Namun referensi gambar yang paling bagus adalah referensi gambar yang telah disiapkan oleh klien. Referensi gambar yang bagus adalah referensi gambar dari berbagai sudut untuk mendapatkan hasil yang akurat.




Gambar 2.2. *Car Blueprint* I
(*Poly-Modeling with 3ds Max Thinking Outside of the Box,* hal. 106)
Contoh gambar referensi sebuah mobil yang diambil dr berbagai sudut.




Gambar 2.3. *Car Blueprint* II
(*Poly-Modeling with 3ds Max Thinking Outside of the Box,* hal. 112)

## 2.3. 3D *Modeling*

Menurut Norman I. Badler dan Andrew S. Glassner dalam materi presentasinya, *3D Object Modeling (1997)* disebutkan bahwa *3D modeling* adalah representasi geometris dari struktur sebuah benda dalam bentuk digital. Untuk membuat sebuah model 3D membutuhkan kombinasi dari tiga hal penting yaitu model 3D, pencahayaan dan *camera view* untuk mencari sudut yang tepat untuk menangkap kesan 3D yang tepat.

Dalam dunia *modeling*, istilah *polygon* tidak asing lagi untuk didengar. Menurut kamus *Alias Maya*, *polygon* adalah sebuah bentuk yang mempunyai beberapa sisi atau bidang yang dibentuk oleh susunan *vertex-vertex* dan *edge* yang dibentuk dari *vertex-vertex* yang saling berhubungan.

Russo (2006) dalam bukunya yang berjudul *Polygonal Modeling: Basic and Advanced Techniques* menjelaskan bahwa dengan *polygon* dapat membuat obyek seperti apapun, bebas memanipulasi bentuknya dengan cara menggabungkan bagian-bagiannya, memotong, menyatukan dengan bangun lain tanpa mengganggu bentuk keseluruhan apabila dikerjakan dengan baik.

Menurut Donati di dalam bukunya *Exploring Digital Cinematography*, secara garis besar, ada beberapa aturan dasar yang digunakan saat kita mengambil sebuah *shoot* yang dapat membantu artis 3D saat membuat hasil *render* yang berkualitas:

1. Menciptakan sudut pandang yang menarik pada saat menampilkan model 3D.

2. Pencahayaan bisa dimulai dengan teknik 3 *point lighting*, dan kemudian dapat dikembangkan sesuai dengan kebutuhan.
3. Parameter rasio pencahayaan sebaiknya sebanding dengan material dari objek tersebut sesuai dengan tekstur permukaan..
4. *Rim light* harus diatur dengan baik sehingga mendukung kesan 3D dari objek.
5. *Background* harus berupa sebuah permukaan yang mempunyai ujung berbentuk melengkung tanpa ujung yang tajam.
6. Ujung bayangan harus lembut ataupun tidak nampak.

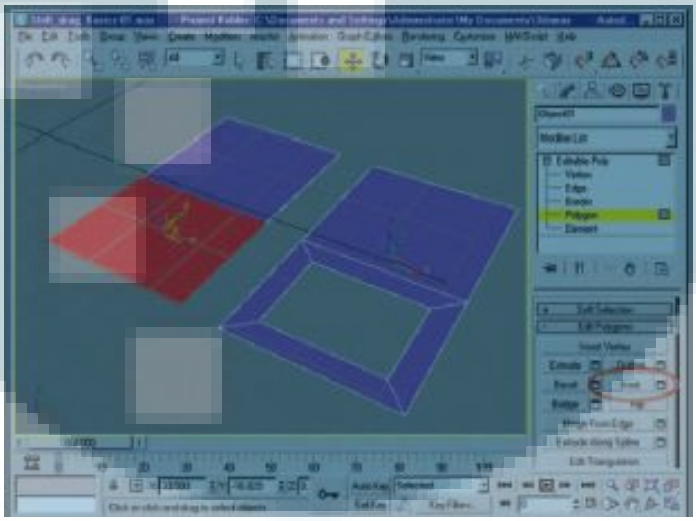
Ada dua tujuan utama untuk membuat model 3D di komputer. Yang pertama adalah membuat *image* suatu benda atau *image making. Image making* membutuhkan model yang terlihat menarik dan mendetail. *Image* tersebut harus mempunyai suatu tingkat kompleksifitas yang menarik untuk dilihat, dan serealis mungkin (bila memang sesuai dengan hal yang ingin dicapai) sehingga bisa meyakinkan orang dan bisa dikategorikan sebagai definisi dari benda yang asli.

Tujuan yang kedua ialah untuk simulasi. Sebuah simulasi membutuhkan model yang akurat untuk mencoba dua benda untuk melihat apakah benda-benda tersebut sudah tepat secara matematis. Misalkan sebuah model pesawat 3D yang harus dibuat dengan akurasi yang tinggi pada semua bagiannya supaya bisa dipelajari sebelum pesawat itu dibuat secara aslinya apakah ada masalah yang akan muncul dan bisa menghindari kesalahan tersebut dengan menguji coba secara digital.

Menurut Daniele Todd dalam bukunya *Poly-Modeling with 3ds Max* (2009), ada beberapa jenis tipe *modeling*:

1. *Polygonal Modeling.*

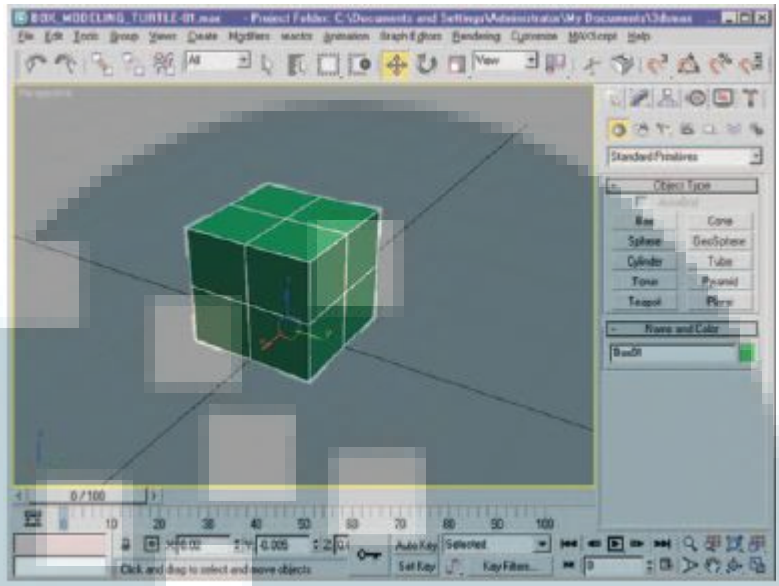
   Banyak model tiga dimensi yang diciptakan sebagai model *polygonal*. *Polygonal Modeling* adalah metode untuk menciptakan model 3D dengan menghubungkan segmen garis melalui titik dalam ruang 3D. Model *polygonal* sangat fleksibel dan dapat di*render* oleh komputer dengan sangat cepat.



Gambar 2.4. *Polygonal Modeling*
(*Poly-Modeling with 3ds Max Thinking Outside of the Box*, hal. 33)

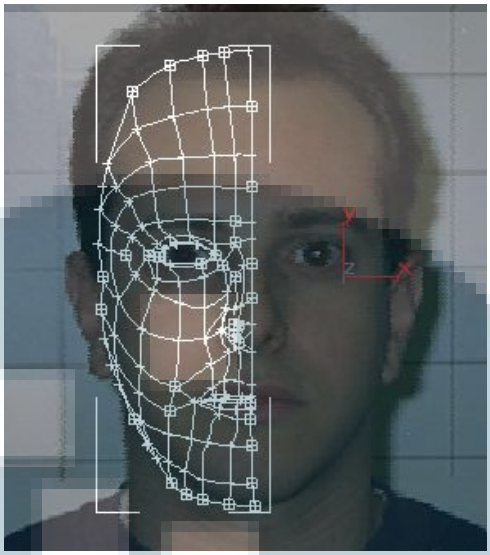
2. *Box Modeling*

   Ini adalah cara paling sederhana dari model objek tiga dimensi. Menggunakan primitif geometris seperti silinder, kerucut, kubus dan bola, model yang kompleks. *Modeling* dengan *box* telah dikenal oleh para 3D *modeler* karena banyak orang yang telah terbiasa dengan sistem kerja *box modeling* dan hampir setiap program 3D memiliki sistem *modeling* ini.



Gambar 2.5. *BoxModeling*
(*Poly-Modeling with 3ds Max Thinking Outside of the Box*, hal. 21)

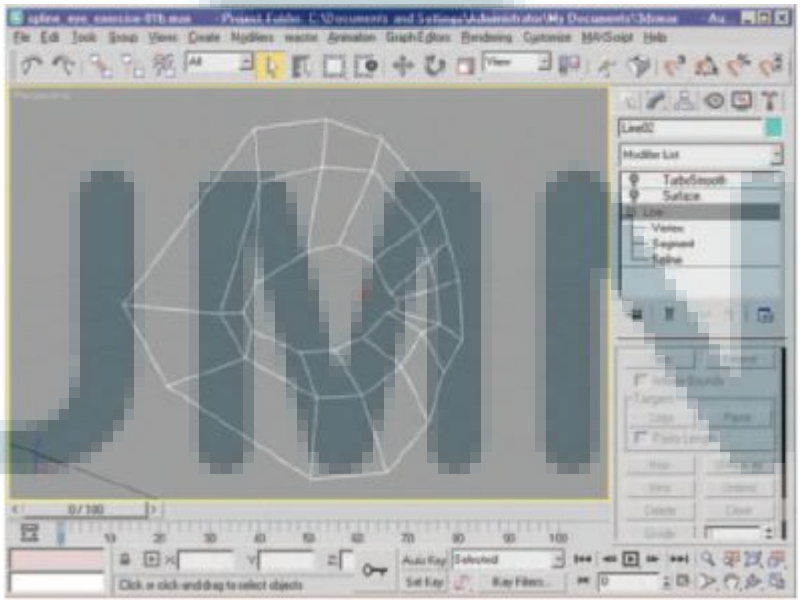
3. NURBS *modeling*. The NURBS *(Non-uniform rational B-spline).* Metode pemodelan ini dapat ditemukan dalam perangkat lunak populer seperti Maya. Pengembang dapat membuat permukaan 3D model yang halus dengan menggunakan teknik pemodelan ini. Tidak seperti teknik *polygonal modeling* yang membutuhkan *polygon* banyak untuk membuat permukaan yang halus, NURBS *modeling* dapat membuat permukaan benar-benar halus.



Gambar 2.6. Modeling Wajah Menggunakan *Spline*
(http://rival13.narod.ru/lessonseng/headmod/FlatSpline-Better.jpg)

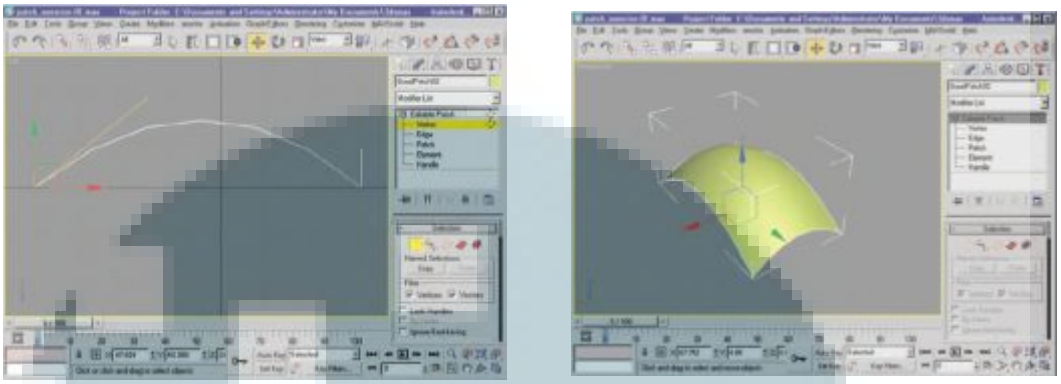
4. *Splines Modeling.*

Metode ini mirip dengan prosedur pemodelan NURBS. Mereka bergantung pada garis lengkung untuk mengidentifikasi permukaan yang terlihat.



Gambar 2.7. *Splines Modeling*
(*Poly-Modeling with 3ds Max Thinking Outside of the Box*, hal. 12)

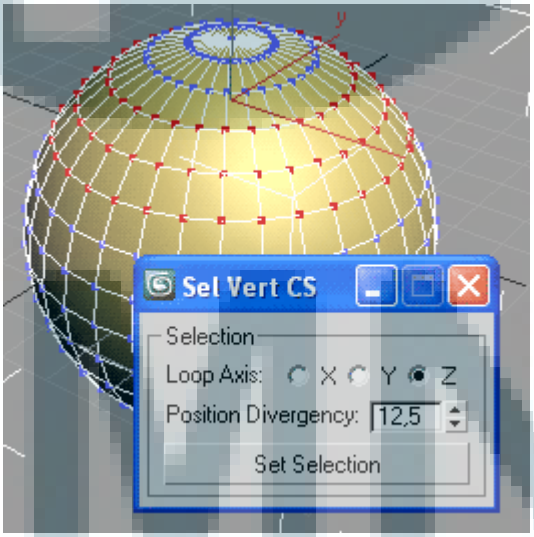
5. *Patches Modeling*



Gambar 2.8. *Patches Modeling*
(*Poly-Modeling with 3ds Max Thinking Outside of the Box*, hal. 15, 16)

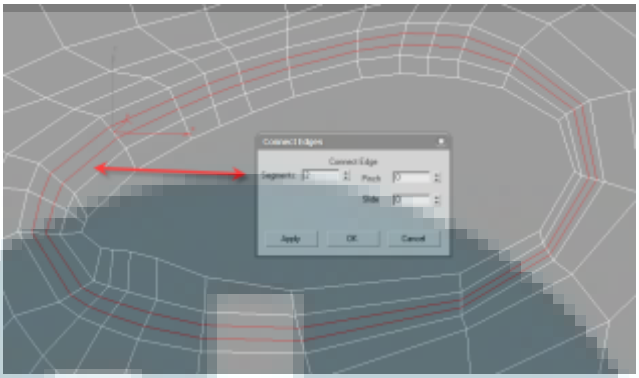
Menurut Andi dalam bukunya *Tips dan Trik 3Ds Max*, dalam berkutat dengan 3D *modeling* pasti akan banyak menggunakan beberapa *subselection* seperti *Vertex, Edge, Border, Polygon,* dan *Element*.

1. *Vertex*, merupakan titik-titik yang menghubungkan dua rusuk atau lebih



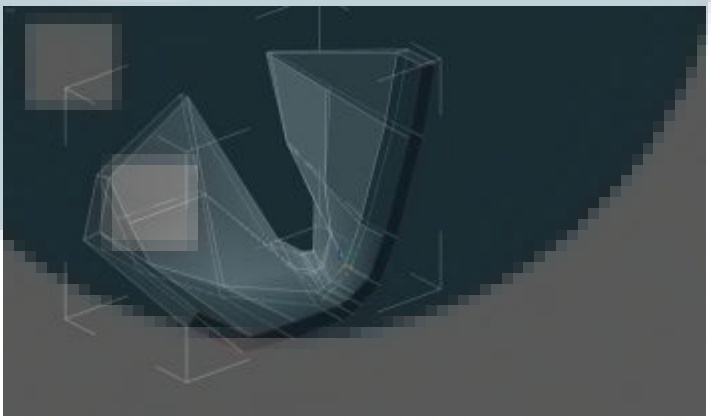

Gambar 2.9. Gambar *Vertex* di Model *3DsMax*
(http://www.scriptspot.com/3ds-max/scripts/select-vertex-cross-section)

2. *Edge*, merupakan rusuk objek. *Edge* menghubungkan dua buah sisi objek.



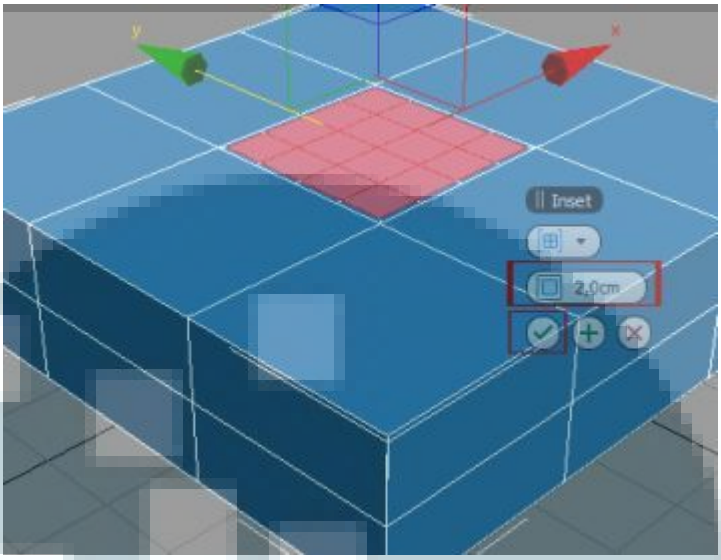
Gambar 2.10. Gambar *Edge*
(http://www.webdesign.org/3d-graphics/tutorials/modeling-an-eye.3365.html)

3. *Border*, merupakan tepi yang mengelilingi area terbuka dari objek. Objek *editable poly* yang seluruhnya tertutup oleh sisi yang tidak memiliki *border*.



Gambar 2.11. Gambar *Border*
(http://www.toppyfly.com/index.php?option=com_content&view=article&id=5:furniture-*modeling*-in-3ds-max&catid=34:tutorials&Itemid=54)

4. *Polygon*, merupakan sisi objek. *Polygon* adalah luasan sisi yang dibatasi oleh lebih dari 3 (tiga) rusuk.




Gambar 2.12. Gambar *Polygon*
(http://www.3dworld-wide.com/3ds-max-create-circular-holes-in-square-objects.html)

5. *Element*, merupakan sekelompok *polygon* tertentu yang biasanya terdapat pada objek *editable poly* yang berasal dari penggabungan 2 (dua) buah objek atau lebih. Objek semacam ini akan terbagi menjadi beberapa *element* berdasarkan objek penyusun asalnya.

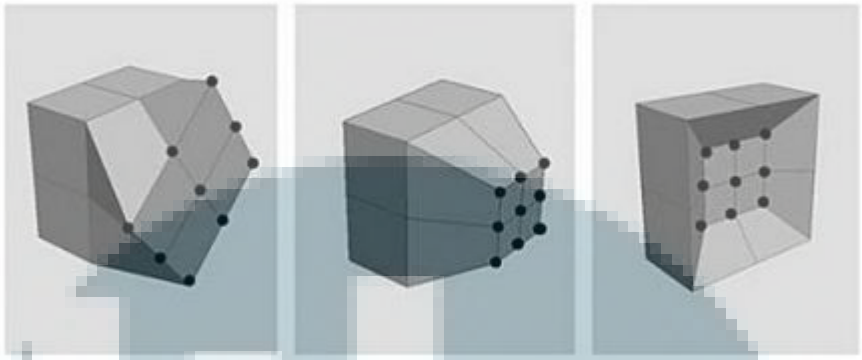

Gambar 2.13. Gambar *Element*
( Dikutip dari *Tips dan Trik 3Ds max* Penerbit Andi hal 26, 2012)

*Polygon Operations*

Sebelum memulai menambah *polygon* dan memotong *polygon* dari sebuah bentuk, sangat penting untuk memahami bagaimana memanipulasi *sub elements* karena setiap modifikasi dan alat yang diterapkan kepada *mesh* akan mempengaruhi posisi atau menambah atau mengurangi jumlah *vertex, polygon* atau *edge* dari *mesh*.

Untuk perintah *Move*, *Rotate*, dan *Scale tools*, dapat dipahami dari nama perintah-perintah tersebut. Setiap perintah biasanya mempunyai kemampuan untuk memanipulasi *gizmo*nya masing-masing. *Move* direpresentasikan oleh tiga arah panah, *rotate* dilambangkan oleh *spherical gizmo,* dan *Scale* oleh *gizmo* yang berbentuk segitiga.

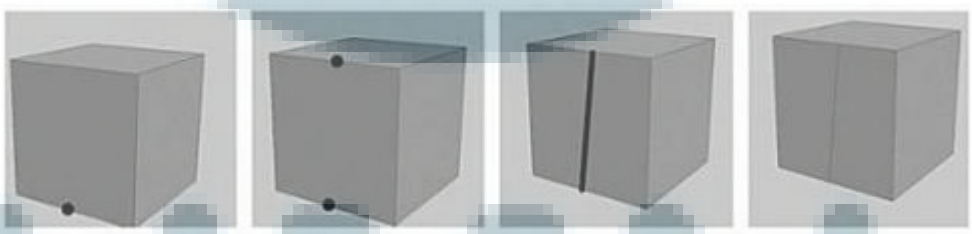
Gambar 2.14. Dari kiri,*Move, Rotate, Scale gizmo*
*(POLYGONAL MODELING: Basic and Advanced Technique,2006)*



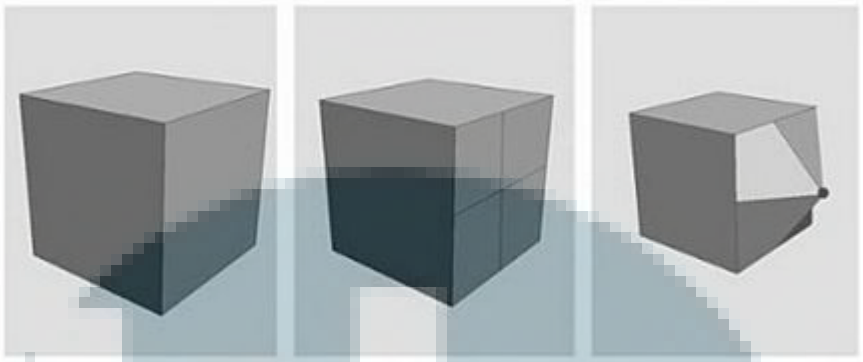
Gambar 2.15. Dari kiri, menggunakan *Rotate tool* dengan *Vertex*, *Scaletool*, dan *Movetool*, menggerakkan *Vertex* yang telah diseleksi mundur *setelah scaling operation*
*(POLYGONAL MODELING: Basic and Advanced Technique,2006)*

*Cutting Polygons*

*Tools Cut* adalah salah satu alat yang paling sering digunakan untuk *polygonal modeling*. Saat memotong atau secara manual membagi atau membelah sebuah *polygon*, anda menambahkan sebduah *Edge.* Setiap *Edge* menambah tekanan kepada *mesh*, dan setiap langkap-langkahnya berbeda secara spesifik kepada tiap aplikasi *modeling*. Gambar di bawah mencontohkan bagaimana penggunaan *Cut.*



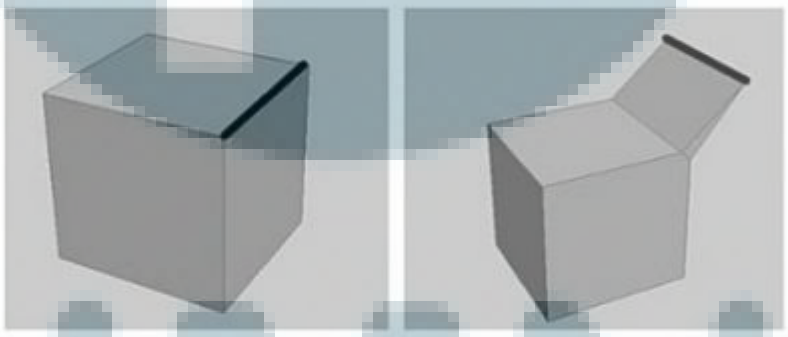
Gambar 2.16. Klik pada titik awal(target potong), lalu klik kearah yang diinginkan untuk membuat garis baru.
*(POLYGONAL MODELING: Basic and Advanced Technique,2006)*



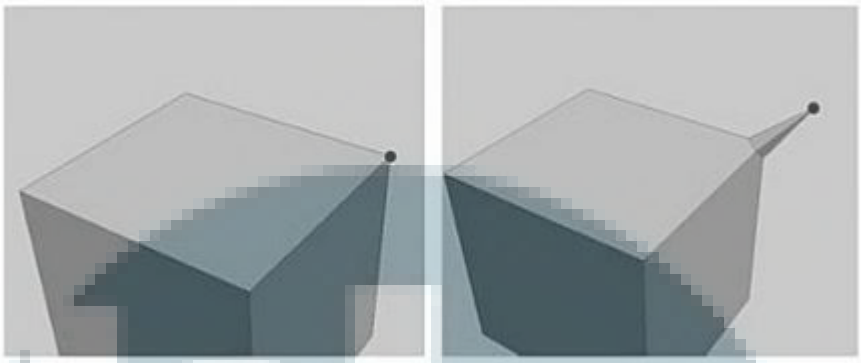
Gambar 2.17. Perpotongan yang didapat dari *Cut tool*, dan *vertex* ditarik ke depan
*(POLYGONAL MODELING: Basic and Advanced Technique,2006)*

*Extruding*

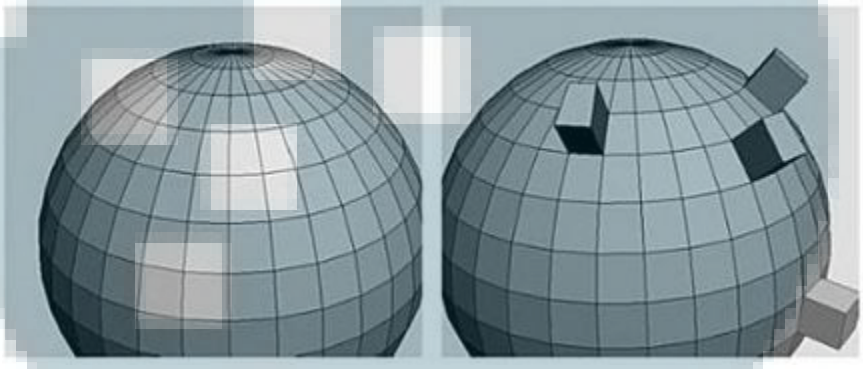
*Extrude tool* dijelaskan di sini sebagai penarikan dari *sub-elements*, seperti *edge*, *face*, *vertex. Face extrusion* berorientasi kepada elemennya yang normal, membuat beberapa *faces* yang baru bersamaan dengan sisi dari *face* asli yang berada *extrusion* dan *extruded face* yang asli.



Gambar 2.18. *Edge Extrusion*
*(POLYGONAL MODELING: Basic and Advanced Technique,2006)*



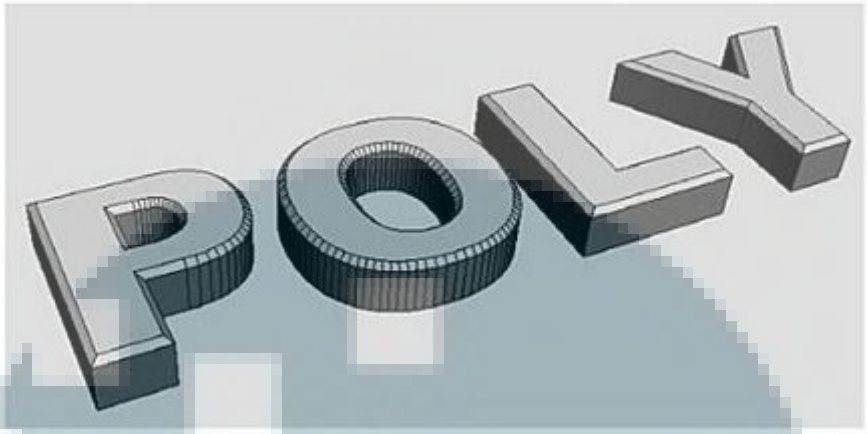
Gambar 2.19. *Vertex Extrusion*
*(POLYGONAL MODELING: Basic and Advanced Technique,2006)*



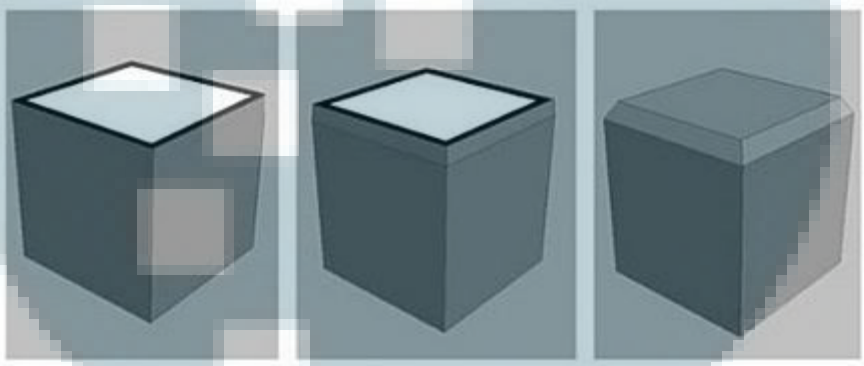
Gambar 2.20. *Face Extrusion*
*(POLYGONAL MODELING: Basic and Advanced Technique,2006)*

*Chamfer/Bevel*

*Chamfer* dan *Bevel* bisa membuat hasil yang sangat mirip. Namun di dalam beberapa aplikasi, fungsinya bisa bervariasi dan dapat memusingkan pengguna.



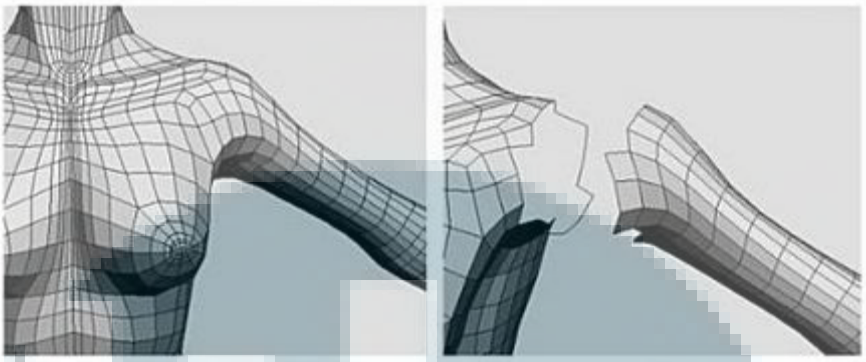

Gambar 2.21. *Text Beveling*
*(POLYGONAL MODELING: Basic and Advanced Technique,2006)*



Gambar 2.22. *Box face Beveling*
*(POLYGONAL MODELING: Basic and Advanced Technique,2006)*

*Attach dan Detach*

*Detach* adalah operasi yang digunakan pada saat ingin memecah *mesh* menjadi dua bagian yang berbeda. Perlu diingat bahwa setiap kali melepas sebuah bagian dari mesh, setiap bagian akan menjadi mesh yang berdiri sendiri dan mempunyai properti, serta tekstur yang berbeda.



Gambar 2.23. *Detaching*
*(POLYGONAL MODELING: Basic and Advanced Technique,2006)*

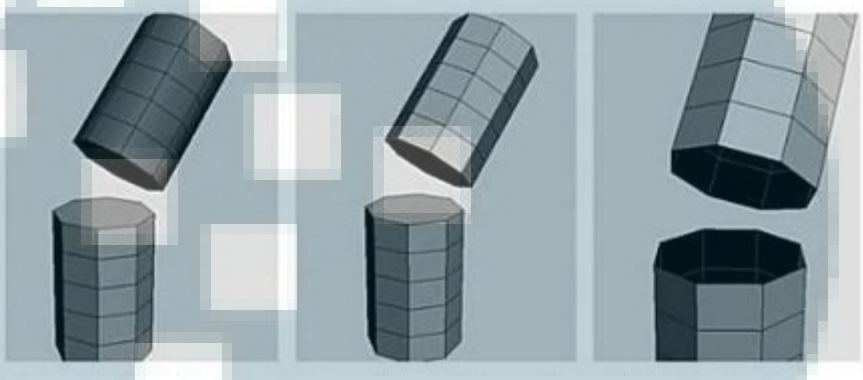
*Attach* adalah operasi yang sangat berguna yang berfungsi untuk menggabungkan dua *mesh* yang berbeda dan menggabungkannya menjadi satu. Saat menggabungkan dua *mesh* yang berbeda, akan ada sebuah mesh yang tidak mempunyai kelanjutan. Perintah *attach* tidak menggabungkan *vertex* secara otomatis harus menggabungkan *vertex* satu demi satu.

Gambar 2.24. *Detaching*
*(POLYGONAL MODELING: Basic and Advanced Technique,2006)*

*Welding*

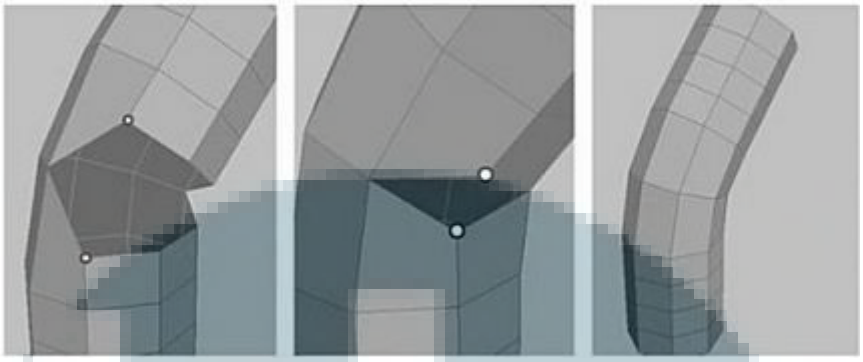
*Welding* adalah alat yang biasa digunakan untuk memperbaiki *mesh* yang mengalami *error* atau menyamakan dua mesh yang berbeda setelah perintah *attach*.

Cara kerja *welding* berbeda-beda pada setiap aplikasi. Di *3Ds Max*, *weld* tidak bisa diaplikasikan pada *vertex* yang tidak dekat dengan sebuah lubang atau pada sebuah *vertex* yang tidak menyabung pada sebuah *edge* melalui *vertex* yang menjadi target. Berbeda pada maya, *welding* dapat dilakukan, namun itu mungkin menimbulkan hasil yang tidak diinginkan pada *mesh* yang ada.



Gambar 2.25. Pada Maya *vertex* dapat di-*weld* dari *cylinder* yang telah di-*attach*, namun pada *3Ds Max cap hole* harus dihapus.
*(POLYGONAL MODELING: Basic and Advanced Technique,2006)*

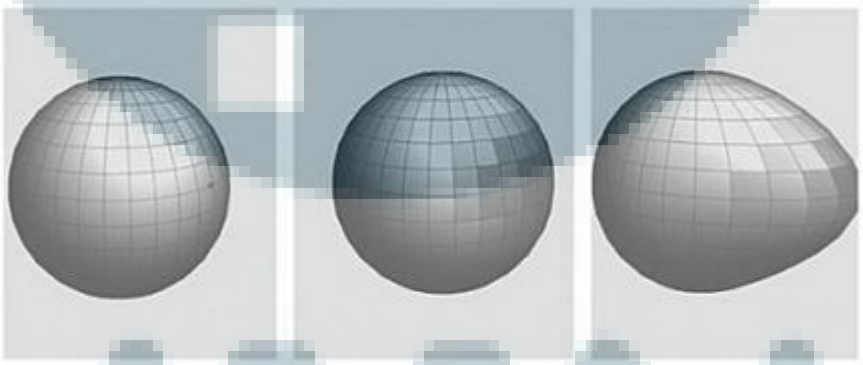
Gambar 2.26. Menggunakan Perintah *Weld*
*(POLYGONAL MODELING: Basic and Advanced Technique,2006)*



Gambar 2.27. *Welding* dua *cylinder* yang mempunyai jumlah sisi yang sama
*(POLYGONAL MODELING: Basic and Advanced Technique,2006)*

*Soft Selection*

Tujuan dari *Soft Selection* adalah untuk memilih *sub selection* dari satu *polygon* untuk kemudian mempengaruhi pada *polygon* di sekitarnya. Misalnya, bila sebuah *vertex* ditarik, maka *vertex* yang berada di sekitarnya akan ikut tertarik bergantung pada sebuah parameter yang ada dalam alat ini.



Gambar 2.28. Menggunakan *Soft selection*
*(POLYGONAL MODELING: Basic and Advanced Technique,2006)*

Menurut Norman I. Badler dan Andrew S. Glassner dalam materi presentasinya *3D Object Modeling* pada halaman 5-6 disebutkan bahwa ada banyak manfaat lain dari model 3D. Di samping ada tujuan jelas seperti bereksperimen dengan sebuah benda secara visual, seperti komputer grafis.

Ada beberapa kegunaan lain yang membentuk sebuah konteks studi yang lebih luas untuk mempelajari model objek. Ada beberapa kegunaan 3D *modeling* untuk dipahami.

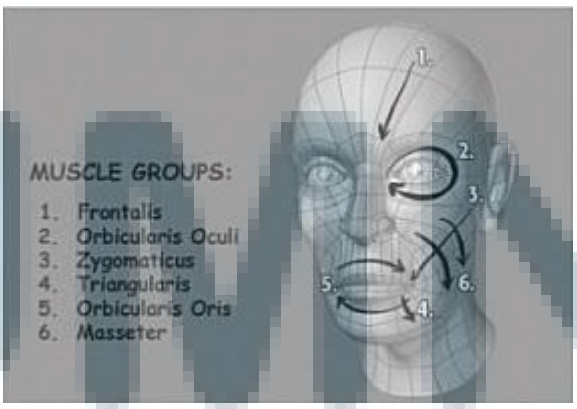
1. Untuk menampilkan bentuk, warna dan desain layout sebuah desain sebelum masuk ke proses
2. Untuk menilai tampilan benda tersebut saat ia berinteraksi dengan lingkungannya
3. Untuk mengamati sinkronisasi hubungan antar bagian
4. Untuk mencari kemungkinan yang akan terjadi bila benda tersebut akan diproses.
5. Untuk menentukan bahan, serta hal lain yang perlu dikurangi atau ditambah. Misal: Saat sebuah bahan sedang diproses perlukah bahan baku ditambah supaya menjadi seperti yang diinginkan, terdapat dua pilihan yaitu mengganti atau tidak mengganti bahan baku agar sesuai .
6. Untuk membuktikan reaksi saat benda atau ojek berinteraksi dengan kejadian disekitarnya. Misalnya bagaimana permukaan objek tersebut berinteraksi dengan cahaya saat cahaya mengenai permukaan benda tersebut: Bagaimana sifat permukaan benda itu (kehalusan, kekasaran, bergelombang) berdasarkan geometri sebuah objek?
7. Untuk memvisualkan algoritma program komputer atau menguji kemampuan visual suatu program.
8. Untuk menunjukan tujuan artistik dari sebuah benda: Apakah antara benda asli dengan objek imajinatif dapat menyampaikan pesan dari artis itu

sendiri? Bagaimana imajinasi dapat digali untuk memvisualkan dunia imajinasi agar lebih menarik?

## 2.4. *Polygonal Flow*

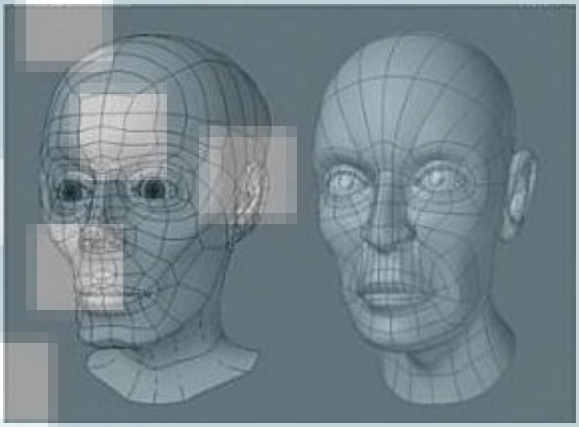
Menurut Gorden, di dalam bukunya *Lightwave 3D[8] Cartoon Character Creation: Modeling and Texturing*, *flow polygon* sangatlah penting dalam membuat sebuah model. Distorsi yang disebabkan oleh *polygon flow* yang jelek dalam pengaturan *subselection* adalah sebab utama yang menjadi masalah di dalam dalam pembuatan model yang rapi.

Masalah dengan kerapian dari permukaan model, ketidakmampuan untuk membuat deformasi yang bagus dengan *morphs* dan *bone*, dan tekstur dari map *UV* yang berdistorsi sering disebabkan oleh *flow polygon* yang tidak rapi. Cara termudah untuk menghindari distorsi *subpatch* adalah dengan mengenali daerah-daerah atau bagian-bagian dari model yang berpotensi untuk menjadi daerah yang bermasalah.




Gambar 2.29. *Polygonal flow* untuk kepala manusia
(Essential Lightwave: The Fastest and Easiest Way to Master Lightwave 3D, hal. 627)

Selama *flow polygon* telah diperhatikan saat awal pembuatan model, banyak distorsi serta *flow* yang buruk bisa dihindari. Dalam membuat sebuah model 3D, *flow polygon* harus dijaga dengan ketat. *Flow polygon* yang rapi akan mengurangi kemungkinan distorsi pada *UV map*, yang memudahkan model untuk dibuat, dan untuk menjaga perubahan bentuk agar rapi.



Gambar 2.30. Perbandingan *Polygonal flow*
(Essential Lightwave: The Fastest and Easiest Way to Master Lightwave 3D, hal. 627)

Ilustrasi di atas menunjukkan *flow polygon* yang buruk (kiri) bisa membuat sudut yang tidak diinginkan dan bagian-bagian yang aneh dan tidak natural, berbeda dengan *flow polygon* yang bagus (kanan) mengikuti alur otot dari model dan membuatnya mudah untuk dikerjakan.

## 2.5. Perbandingan Jumlah *Polygon*

Di dalam buku Profesional *3Ds Max* (2011) dikatakan bahwa *low poly* dan *high poly* adalah istilah yang biasa digunakan untuk menggambarkan jumlah *polygon* (atau *triangles*) yang membentuk suatu model 3D.

Menurut Thorn, di dalam bukunya *Game Development Principles* dikatakan bahwa tidak ada patokan baku mengenai berapa jumlah *polygon* yang harus dimiliki suatu model untuk masuk ke dalam kategori low atau *high poly*, hanya saja secara umum model *low poly* memiliki *polygon* yang jauh lebih kecil daripada model *high poly*, biasanya di kisaran ribuan atau 4 digit. Sedangkan model *high poly* bisa mencapai puluhan bahkan ratusan ribu *polygon* dalam satu model.

Thorn juga berkata bahwa 500 *polygon* untuk sebuah properti yang penting didalam sebuah game, seperti senjata dikategorikan sebagai *lowpoly* menurut standar beberapa *game developer*. Menurut Andrew Paquette dalam bukunya *Computer Graphics for Artists II: Environments and Characters*, untuk sebuah bangunan 3D ada beberapa tingkat resolusi yang bisa dikelompokkan:

1. Lebih sedikit daripada 500 *polygon*: *Extremely low resolution*
2. 501-2000 *polygon*: *Low resolution*
3. 2001-5000 *polygon*: *Medium resolution*
4. 5001- 15,000 *polygon*: *High resolution*
5. 15,001- 50,000 *polygon*: *Very high resolution*
6. Diatas 50,000 *polygon*: *Extremely high resolution*

Dalam membuat sebuah model 3D, jumlah *polygon* akan mempengaruhi beberapa hal:

1. Kualitas model saat di*render* dalam *render engine*.

   Jumlah *polygon* yang lebih banyak akan membuat sebuah model terlihat lebih 'mulus' dibandingkan model dengan jumlah *polygon* lebih rendah.

2. Performa pada *engine* saat me*render* model.

   *Rendering* adalah proses menampilkan model 3D ke layar dalam sebuah aplikasi atau *game*, dengan cara mengolah informasi yang dimiliki model-model 3D dan informasi pendukungnya (sumber cahaya, efek kamera, dan lain-lain) untuk menghasilkan tampilan 3D yang meyakinkan.

   Proses *rendering* ini bukan hanya memproses per model, bahkan pada dasarnya akan dilakukan per permukaan *polygon*. Karena itu, semakin banyak *polygon* yang digunakan, proses *rendering* pun akan semakin lambat.

Menurut Alan Wyatt dalam bukunya 3D *Game*s: Real-Time *Rendering* and Software Technology, Volume 1 (2001), *Game* umumnya menggunakan teknik *real-time rendering*, yaitu proses *rendering* dilakukan tiap *frame*, atau sekitar 30 kali per detik. Dengan teknik ini, proses *rendering* akan sangat mempengaruhi performa *game*, walaupun dalam *game* masih banyak proses yang perlu dilakukan, misalnya *collision detection, game logic*, AI, dan lain-lain

Berbeda dengan animasi untuk film atau *cutscene* (adegan) *in-game* yang umumnya menggunakan teknik *pre-rendering*, yaitu me-*render* ke dalam suatu *file* terpisah yang nantinya akan ditampilkan ke pengguna dalam bentuk *film*.

Dengan menggunakan *pre-rendering*, pengguna hanya akan melihat hasil akhir sesuai dengan jumlah waktu adegan yang bersangkutan, tanpa dipengaruhi prosesnya, padahal sebuah film 3D pendek sekitar 5 menit saja bisa membutuhkan waktu *pre-rendering* selama berjam-jam.

Model *high poly* akan membutuhkan waktu yang lebih lama untuk di*render*, karena jumlah *polygon*nya lebih banyak, lebih banyak pula waktu yang dibutuhkan untuk memproses semua *polygon* tersebut. Model seperti ini lebih cocok untuk segala sesuatu yang membutuhkan detail dan kualitas yang sangat tinggi.

Sedangkan model *low poly* akan jauh lebih cepat di *render*, dan untuk pemrosesan seperti *collision detection*, *physics simulation*, dan proses-proses lain yang membutuhkan informasi model 3D dalam *game*, akan membutuhkan waktu yang lebih cepat dibandingkan model *high poly*, sehingga lebih umum digunakan dalam *game*.

Walaupun demikian, tetap perlu disesuaikan dengan kebutuhan. Bila tidak terlalu membutuhkan pengolahan informasi model 3D, misalnya aplikasi *point and click*, dimana jika sebuah *trigger* ditekan maka akan dijalankan suatu animasi yang sudah ditentukan, maka bisa menggunakan model *high poly*.

Tapi jika *game* akan mengolah informasi model 3D tersebut sebagai bagian dari *game play*, misalnya *game First Person Shooting*, dimana tiap peluru akan diperiksa apakah mengenai seorang pemain, dan titik tabrakan model pemain dan peluru akan mempengaruhi jalannya permainan (apakah akan terjadi *headshot*, mengenai tangan sehingga pemain tidak bisa menggunakan senjata,

atau mengenai kaki sehingga pemain menjadi lebih lambat) akan jauh lebih menguntungkan jika model *low poly* yang digunakan.

Ada beberapa hal yang bisa dilakukan untuk membuat kualitas *game* yang lebih bagus dari sisi grafis walaupun menggunakan model *low poly*. Beberapa *platform* (terutama *console* dan *PC*, karena memiliki *hardware* lebih canggih) memiliki fitur-fitur yang dapat digunakan:

1. *Bump mapping* dan normal *mapping*.

   *Bump map* dan normal *map* adalah suatu file gambar yang bisa digunakan sebagai informasi tambahan selain file tekstur dalam me*render* sebuah model 3D.

   Kedua file ini akan digunakan sebagai informasi kedalaman yang akan diperhitungkan dalam me*render* bayangan di suatu permukaan objek. Dengan teknik ini, efek yang dibuat seperti relief pada tembok batu menggunakan model tembok yang sebenarnya datar.

2. *Post processing* menggunakan *shader*.

   *Shader* adalah teknologi yang digunakan dalam proses *graphic rendering* secara umum, tapi penggunaannya luar biasa banyak (sebenarnya b*ump mapping* dan *normal mapping* adalah contoh implementasi *shader* yang sudah di*handle* di tingkat *rendering library* atau bahkan pada tingkat *hardware*) bisa membuat efek-efek tembahan menggunakan *shader*, misalnya membuat efek *motion blur, bloom*, dan masih banyak lagi. Tapi untuk menggunakan *shader* ini, harus mempelajari bahasa *shader*,

biasanya HLSL atau GLSL, dan konsepnya cukup berbeda dari bahasa pemrograman biasa.

## 2.6. Tekstur dan Material Objek

Dalam buku *How to Cheat in 3Ds Max*, dikatakan bahwa *texturing* bisa dilakukan dengan cara yang mudah, berbeda dengan tahun 1990 dimana saat itu komputer grafis masih baru dan *modeling* merupakan kemampuan yang jarang dimiliki orang.

Sekarang banyak klien mengharapkan semua sangat mirip seperti aslinya di dunia nyata. Penggunaan objek 3D dapat mempersingkat waktu dengan mengumpulkan banyak referensi di lapangan deperti foto, gambar, video dan tekstur-tekstur benda yang akan dibuat.

*Material* sebuah objek menampilkan sifat dan karakteristik bahan objek tersebut sebagai representasi kondisi alamiahnya dalam dunia nyata yang membedakan objek tersebut dengan objek lain. Misalnya, material kayu berbeda dengan material logam.

*Map* sebuah material merupakan penggambaran kondisi tekstur dari permukaan objek. Misalnya, objek kayu sengon berbeda dengan objek kayu jati, meskipun keduanya sama-sama bermaterial kayu. Yang membedakan keduanya adalah tekstur permukaan, warna, dan beberapa karakteristik lainnya

### 2.6.1. Warna

Sifat permukaan, seperti *diffuseness,* refleksisifitas, dan lain-lain direpresentasikan dengan sebuah nilai. Nilai ini menentukan sifat dari parameter-parameter tersebut.

Misalnya pada *roughness*, makin besar nilai parameternya, makin kasar objek tersebut.

Misalkan sebuah keranjang yang berisi buah apel yang mempunyai permukaan yang mengkilap. Saat keranjang tersebut disinari dengan cahaya *spotlight* biru terang, semua buah apel akan berubah menjadi ungu dikarenakan perpaduan warna antara *highlight* biru dari cahaya dengan warna merah di permukaan buah apel. Jadi, meskipun *material* apel diubah menjadi warna merah, hasil akhirnya pun akan sangat berbeda karena cahaya membuat warna berubah.

Di dalam *material editor* ada beberapa *color swatches* yang mengontrol beberapa aspek yang berbeda dalam warna sebuah objek. Beberapa poin di bawah ini menggambarkan tipe-tipe *color swatches* yang tersedia untuk macam-macam *material*.

1. *Ambient*: Menggambarkan latar belakang keseluruhan *lighting* yang berpengaruh kepada semua objek di dalam *scene*, termasuk warna dari sebuah benda saat benda tersebut berada di dalam bayangan. Warna ini terkunci kepada *Diffuse Color* secara otomatis supaya saat mereka diubah mereka berubah bersamaan.
2. *Diffuse*: Permukaan warna dari sebuah objek di saat normal, penuh dengan cahaya putih. Warna normal sebuah benda biasanya ditentukan oleh *Diffuse Color*nya.
3. *Specular*: Warna dari *highlights* dimana cahaya difokuskan kepada permukaan objek yang berkilau.

4. *Self-Illumination*: Warna yang ditimbulkan dari dalam objek. Warna ini menghilangkan semua cahaya dari objek.
5. *Filter*: Warna yang timbul dikarenakan oleh cahaya yang bersinar melalui objek yang transparan.
6. *Reflect*: Warna yang dipantulkan oleh material *raytrace* kepada semua objek di *scene*.
7. *Luminosity*: Menyebabkan sebuah objek untuk bersinar dengan warna yang menyebar. Ini mirip dengan *Self-Illumination Color* namun bisa mengikuti *Diffuse Color*nya masing-masing.

### 2.6.2. *Opacity* dan *Transparency*

Objek yang tidak tembus cahaya adalah benda yang tidak dapat dilihat secara tembus pandang, seperti batu dan pohon. Di lain tangan benda yang tembus pandang seperti kaca, dan plastik bening. Material di *3Ds Max* termasuk beberapa kontrol untuk mengatur properti tersebut, termasuk *Opacity* dan beberapa pengaturan *Transparency* lainnya.

*Opacity* adalah suatu jumlah yang dimiliki suatu objek untuk menolak maupun menerima presentase cahaya untuk dapat menembus benda tersebut. Sebuah objek dengan 0% *Opacity* benar-benar tembus pandang, dan sebuah benda dengan 100% *Opacity* tidak membiarkan cahaya apapun menembusnya.

*Transparency* ialah sejumlah cahaya yang dibiarkan menembus permukaan sebuah objek. Karena ini berkebalikan dengan *Opacity*, *Transparency* bisa diukur melalui nilai *opacity*. Beberapa pilihan membuat *Transparency*, termasuk *Falloff*, *Amount*, dan *Type* dapat diatur.

### 2.6.3. *Reflection* dan *Refraction*

Sebuah pantulan terjadi saat benda dihadapkan ke sebuah cermin. Benda-benda yang bersinar memberi pantulan tentang sekeliling benda tersebut. Dengan mengatur sebuah nilai pantul material sebuah benda, anda dapat mengatur juga seberapa besar ia memantulkan sekelilingnya. Sebuah cermin misalkan, memantulkan semuanya, tapi berbeda dengan batu yang tidak memantulkan apapun.

*Reflection Dimming* mengatur seberapa banyak pantulan asli dari sebuah objek mulai menghilang saat di sekelilingnya mulai terpantulkan didalam *scene*. *Refraction* adalah pembengkokan cahaya saat melalui material yang tembus pandang. Jumlah dari pembiasan yang diproduksi digambarkan sebagai sebuah nilai yang disebut *Index of Refraction value.* Nilai standar dari *Index of Refraction* adalah jumlah cahaya yang membiaskan saat melalui atau menembus benda transparan. Sebagai contoh, sebuah berlian membengkokkan cahaya lebih dari sebuah gelas yang berisi air, jadi berlian tersebut mempunyai *Index of Refraction value* yang lebih besar.

Nilai bawaan *Index of Refraction value* adalah 1.0 untuk objek yang sama sekali tidak membengkokkan cahaya. Air mempunyai nilai 1.3, gelas bernilai sekitar 1.5, dan kristal padat mempunyai nilai sekitar 2.0.

Gambar 2.31. Gambar *Reflection*
(http://www.dmmultimedia.com/schnorb_lighting/max9_paint_004.jpg)

Gambar 2.32. Gambar *Refraction*
(http://cienel.net/images/tutorial-images/44/28.jpg )

### 2.6.4. *Shininess* dan *Specular Highlights*

Objek yang berkilau, seperti *polished metal* atau jendela yang bersih, mempunyai *highlights* dimana cahaya memantul dipermukaan. *Highlights* ini disebut *Specular Highlights* dan ditentukan oleh *Specular Setting. Setting* ini termasuk *Specular Level, Glossiness, and Soften Values.*

*Specular level* adalah sebuah pengaturan untuk mengatur intensitas dari sebuah *Highlight*. *Glossiness* menentukan ukuran dari *Highlight*: *Glossiness* yang semakin tinggi akan menyebabkan *Highlight* yang semakin kecil. Nilai yang semakin halus akan mempertipis *Highlight* dengan mengurangi intensitas dan menambah ukurannya.

Material yang kasar mempunyai ciri khas yang berkebalikan dengan material yang berkilau dan hampir tidak ada *Highlight.* Pengaturan kekasaran atau keburaman membuat seberapa cepat warna membaur dengan warna ambien. Material baju dan kain mempunyai *Roughness Value* yang besar, sedangkan plastik dan besi sebaliknya.

( Dikutip dari *3DsMax Bible*)

### 2.6.5. *Normal Map*

Menurut panduan pengguna dari *3Ds Max*, sebuah *normal* adalah sebuah *vector unit* yang menentukan ke arah manakah sebuah *vertex* atau *face* menghadap. *Normal map* adalah sebuah map tekstur yang digunakan program untuk menghasilkan *normal* dari setiap *pixel* untuk bidang dari *low-resolution model* (model dengan jumlah *poly* yang sedikit).

*Map* ini menentukan sebanyak apa cahaya yang diterima per *pixel* dari sumber cahaya di dalam *game* yang nantinya akan menimbulkan ilusi kedalaman dan detail. Detail dari versi *high poly model* akan dipindahkan ke *normal map* yang nantinya diaplikasikan pada *low poly* model dengan menggunakan *Render to Texture* pada *3Ds Max* .

### 2.6.6. *UVW Map*

Menurut Franson dan Thomas (2007), sebuah model terdiri dari *vertex* dalam jumlah yang sangat banyak. Dalam ruang 3D, sebuah *vertex* memiliki koordinat X, Y, dan Z yang menentukan lokasinya. Saat sebuah *mesh* dibuat, *mesh* tersebut akan membuat sebuah set *vertex* duplikat yang tak terlihat yang disebut dengan koordinat tekstur, atau *UV*'s. Huruf U dan V (kadang W) sebenarnya sama dengan X, Y, dan Z. Setelah model objek yang diinginkan selesai, barulah dibuat *map* tekstur *UV* sehingga dapat menggambarnya di program seperti *Photoshop* dengan menggunakan map tersebut.

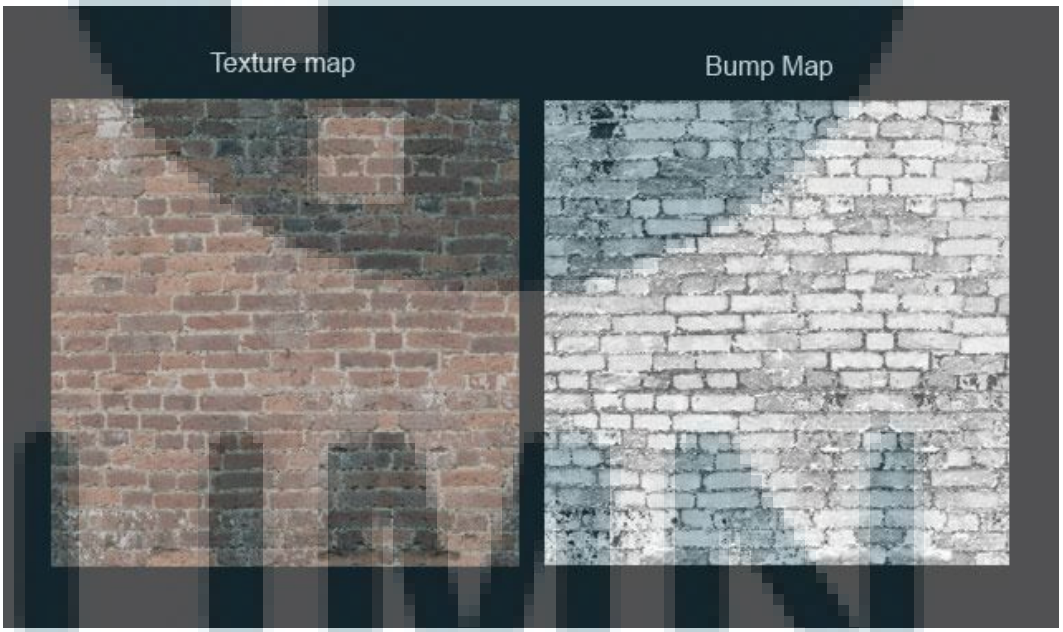
Hubungan antara setiap titik *UV* dengan sebuah *pixel* di dalam sebuah *image map* didefinisikan secara eksplisit. Lebih dari itu, tiap dari titik *UV* diubah menjadi *vertex* dari model , dan hubungan antara *UV's* dan gambar sejalan dengan *map* kepada model 3D. Sebuah *UV map* akan nampak seperti model yang diratakan dan ditumpangkan kepada sebuah *image map.*

Jadi *UV* bisa digunakan sebagai panduan untuk menggambarkan *image map*. Terlebih lagi *UV map* bisa di*edit*, ini berarti pengguna dapat mengedit *pointer UV point* (mengatur dan memindahkan) *vertex* dan *polygon* secara bebas sesuai kebutuhan. Itu adalah alasan mengapa metode ini cukup menghabiskan waktu.

### 2.6.7. *Bump Map*

Menurut panduan pengguna *3Ds Max*, sebuah *bump map* adalah sebuah *map* yang menggunakan gambar *grayscale* untuk menimbulkan ilusi kedalaman seperti *emboss*. Kedalamannya bergantung pada intensitas warna *grayscale*. Semakin hitam maka area tersebut akan makin menjorok kedalam dan sebaliknya.

Kedalaman pada *bump map* memiliki batas, apabila ingin membuat kedalaman yang ekstrim, lebih disarankan menggunakan teknik *modeling*. Menurut Franson & Thomas (2007), kelemahan dari *bump map* adalah pada model yang telah di*render*, arah cahaya nampak datang hanya dari satu arah, biasanya dari atas.




Gambar 2.33. Contoh *Bump Map*
(http://3dmodeling4business.com/wp-content/uploads/2012/10/Bump-Map-2.jpg)

### 2.6.8. *Unwraping*

Ada dua metode dasar untuk meng-*unwrap* sebuah *UV* dari sebuah model yaitu *flatenning* dan *unwrapping*. Salah satunya adalah dengan memilih tipe proyeksi yang cocok atau sesuai dengan model objek, dan kemudian mengubah proyeksi itu menjadi *UV map*, kemudian menyesuaikan *image map* sesuai kebutuhan.

Sebuah map tekstur berbentuk 2D dimana X adalah koordinat horizontal dan Y adalah vertikal. Dalam *3Ds Max*, pembuatan map *UV* dilakukan dengan cara memotong koordinat tekstur dan memproyeksikannya dalam bidang datar, yang dapat dianalogikan sebagai memotong kaos pada garis jahitannya dan meletakkannya dalam bidang datar. Karena terletak dalam bidang datar, *vertex* tersebut tidak lagi terletak dalam ruang 3D, tidak memiliki dimensi ketiga, atau nilai V. Maka itu disebut dengan istilah *UV*.

Terkadang sebuah *vertex* bisa tertumpuk lalu menyebabkan *error*. Untuk mengecek *error* semacam ini, bisa menggunakan *checker map*. Sebuah *mesh* yang telah dilakukan *unwrap* dengan rapi akan menampilkan *checker map* yang rapi .

Gambar 2.34. Contoh *Unwrapping* Sebuah Karakter 3D
( *Texturing : Concept and Techniques, (2004))*

## 2.7. Pencahayaan

Menurut Guntur pada bukunya *3D Realistic Project: Bedroom,* model dari pencahayaan, dipakai untuk menghitung intensitas dari cahaya yang terlihat dari setiap posisi pada setiap permukaan benda yang terlihat oleh kamera. Ketika melihat sebuah benda, terlihat cahaya yang dipantulkan dari permukaan benda, dimana cahaya ini merupakan integrasi dari sumber-sumber cahaya serta cahaya yang berasal dari pantulan cahaya permukaan-permukaan yang lain.

Karena itu benda-benda yang tidak langsung menerima cahaya dari sumber cahaya, masih mungkin terlihat bila menerima cahaya pantulan yang cukup dari benda di dekatnya. Model sederhana dari sumber cahaya adalah sebuah titik sumber, dimana dari titik ini cahaya dipancarkan. Perhitungan pencahayaan bergantung pada sifat dari permukaan yang terkena cahaya, kondisi dari cahaya latar serta spesifikasi sumber cahaya.

Semua sumber cahaya dimodelkan sebagai sumber titik yang dispesifikasikan dengan :

1. Lokasi:

   Lokasi (x,y,z) dari sebuah sumber cahaya akan menentukan pengaruhnya terhadap sebuah objek.

2. Intensitas:

   Intensitas cahaya menyatakan kekuatan cahaya yang dipancarkan oleh sebuah sumber cahaya. Parameter ini merupakan angka, yang biasanya makin besar nilainya, makin terang sumber cahaya tersebut.

3. Warna:

   Warna cahaya dari sumber ini akan mempengaruhi warna dari sebuah objek, jadi selain warna objek tersebut warna cahaya yang jatuh pada objek tersebut akan mempengaruhi warna pada *rendering*. Warna cahaya ini biasanya terdiri dari 3 warna dasar grafika komputer, yaitu: merah, hijau, biru atau lebih dikenal dengan RGB.

Di dalam bukunya *Exploring Digital* Cinematography yang dikarang oleh Donnati, artis 3D sering menghabiskan waktu yang cukup banyak, disaat memperlihatkan *demo reel* ataupun portofolio mereka. Kebanyakan pencahayaan yang ada lebih menjatuhkan daripada membantu. *Lighting* yang buruk akan membuat model serta tekstur yang bagus akan nampak datar dan tidak menarik.


Gambar 2.35. Pencahayaan yang Buruk.
(*Product Design Graphics with Materials Technology*, hal 67)


Gambar 2.36. Penchayaan yang Baik.
(*Product Design Graphics with Materials Technology*, hal 67)

### 2.7.1. *Standard Lights*

Meskipun teknik *radiocity* mampu menciptakan hasil yang indah dan sangat realistis, artis *lighting proffesional* akan sering bekerja menggunakan lampu standar, untuk beberapa alasan. Alasan utama di balik menggunakan lampu standar yang dapat dengan mudah dikontrol dan disesuaikan untuk menghasilkan setiap gaya pencahayaan dan pengaturan *standard light* memungkinkan seniman untuk mempercepat kinerja mereka, terutama pada waktu *render*.

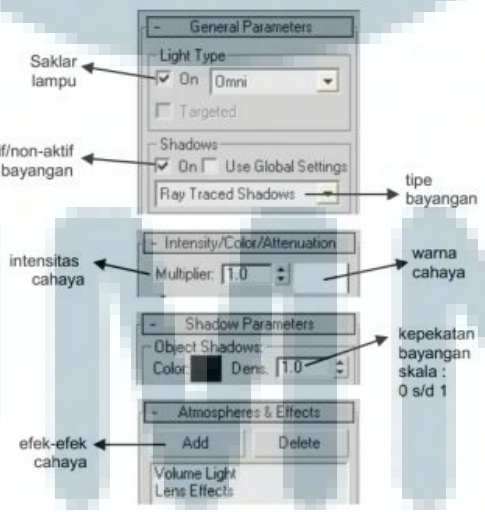
Menurut bentuk cahayanya, *lights* di *3Ds Max* dibagi menjadi beberapa jenis, yaitu :

a. *Omni Lights*

Meskipun disebut sebagai lampu omni oleh pengguna *3Ds Max*, pengguna 3D lainnya menyebutnya lampu spot atau lampu radial. Nama ini membantu menjelaskan bagaimana cara kerja omni karena lampu ini memberikan titik sumber pencahayaan yang keluar secara radial dari satu titik yang lebih kecil.

Karena secara *omni directional* lampu ini mendistribusikan pencahayaan mereka melalui ruang,mereka adalah cahaya yang paling mudah untuk diatur.

Gambar 2.37. Arah cahaya Omni Light
(http://www.mahanani.web.id/2013/03/lightingpencahayaan-3d-max.html)



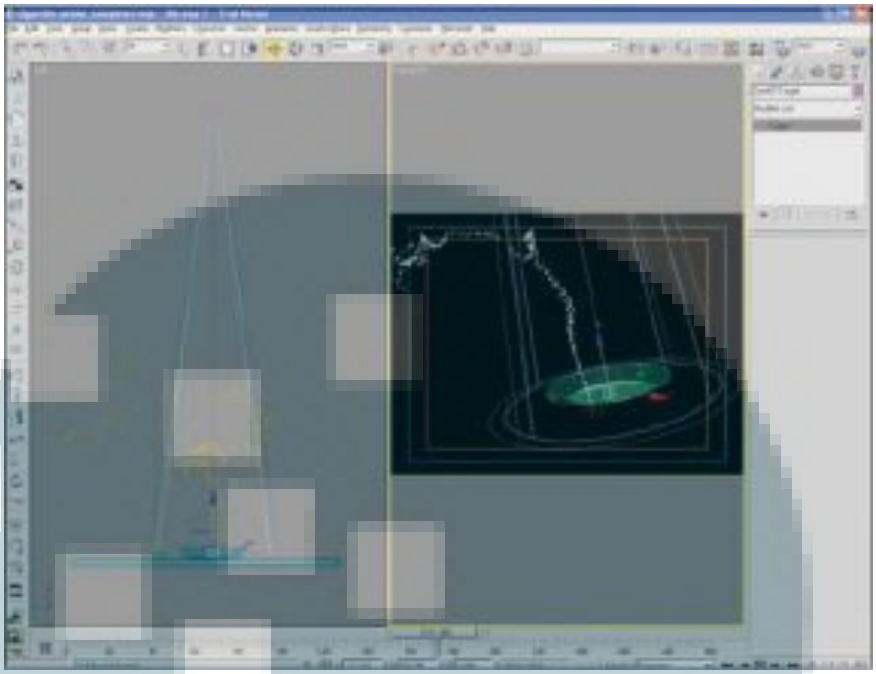

Gambar 2.38. Parameter – Parameter yang Penting di Dalam *Omni Light*
(http://www.mahanani.web.id/2013/03/lightingpencahayaan-3d-max.html)

b. *Spots Lights*

*Spots* merupakan dasar dari banyak situasi pencahayaan. Hal ini disebabkan oleh fakta bahwa sorotan yang dimunculkan oleh *Spots* dapat dikontrol dalam hal arah sesuai dengan keinginan. Tempat melemparkan cahaya sinar terfokus seperti sinar senter.

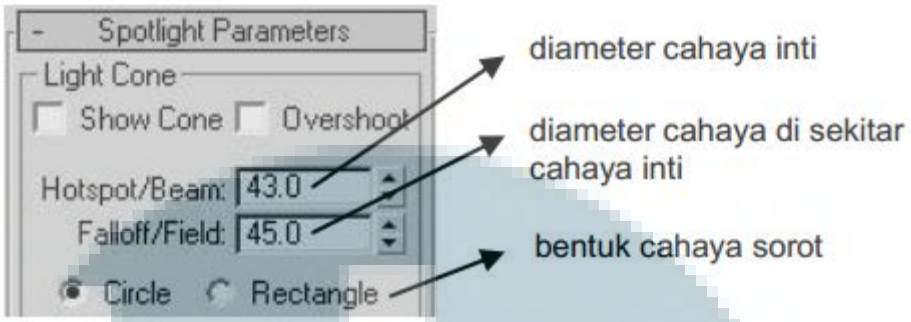
Seperti lampu *omni*, *Spots* memancarkan cahaya dari satu titik tunggal yang sangat kecil, tapi tidak seperti lampu *omni*, pencahayaan berbentuk kerucut. Dengan demikian, lampu sorot perlu diarahkan dan ini dapat dilakukan dengan beberapa cara. Salah satu metode adalah dengan mengatur posisi cahaya sesuai keinginan dan putar sampai itu diarahkan dengan benar, yang mungkin bukan merupakan metode yang paling sederhana, tetapi mungkin yang paling medekati dengan bagaimana cahaya dunia nyata.

Sebuah metode yang sering membuktikan dirinya untuk menjadi lebih berguna mencakup pemberian cahaya kepada objek target, sehingga tetap menghadapi ini tidak peduli bagaimana cahaya berorientasi.



Gambar 2.39. *Spot Lights* 1
(Essential CG Lighting Techniques with 3ds Max 3rd Edition, hal 24)


Gambar 2.40. Contoh gambar *Spot Lights*2
(http://www.mahanani.web.id/2013/03/lightingpencahayaan-3d-max.html)




Gambar 2.41. Parameter yang ada di *Spot Lights*
(http://www.mahanani.web.id/2013/03/lightingpencahayaan-3d-max.html)

*Spot Light* akan memainkan peran utama dalam desain skema pencahayaan banyak karena sifat lampu ini yang mudah dikendalikan. Fakta bahwa *spot light* dapat dengan mudah ditargetkan, diberi *fall off* yang dapat berkisar dari samar-samar dan keras ke lembut dan halus, membuat *spot light* pilihan yang jelas untuk besar jumlah tugas pencahayaan.

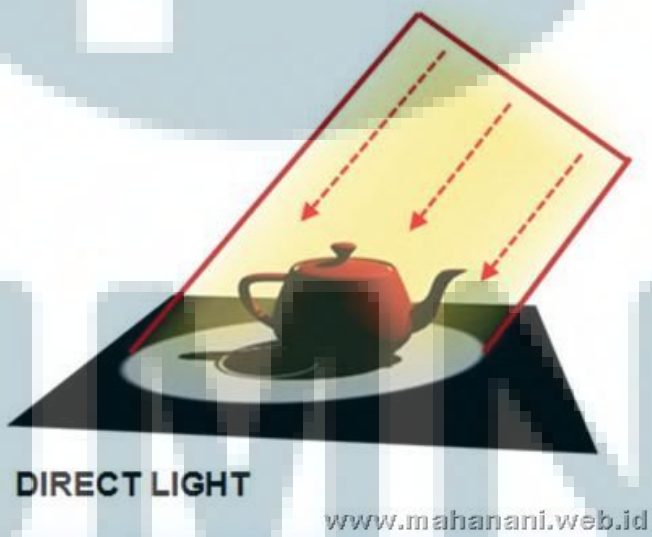
c. *Direct Lights*

Jika menempatkan omni yang relatif dekat dengan satu atau lebih obyek, akan terlihat bahwa bayangan oleh benda akan tergantung pada kedudukan relatif mereka ke sumber cahaya. Bila anda memindahkan cahaya ke arah yang lebih jauh maka Anda akan melihat bayangan menjadi semakin sejajar. Pindahkan sumber ini jarak tak terbatas jauhnya dan sumber cahaya akan dilemparkan sinar paralel.

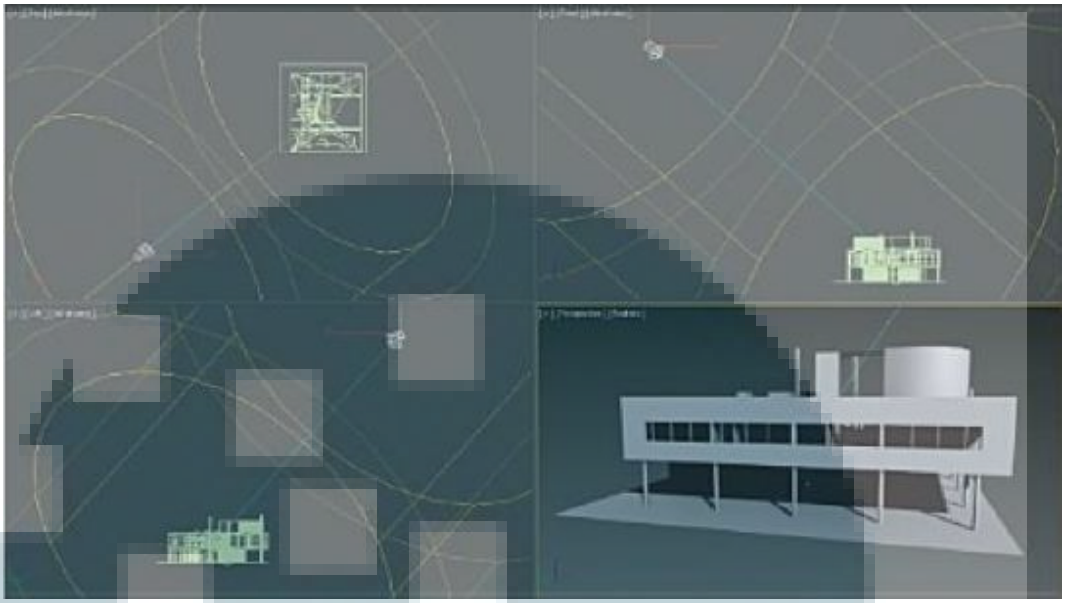
Bila matahari tidak cukup jauh untuk menebarkan cahaya benar-benar paralel. Ini kemudian adalah peran *direct light* untuk melemparkan sinar cahaya paralel dalam satu arah. Hal ini tidak mengherankan kemudian bahwa lampu ini digunakan terutama untuk mensimulasikan sinar matahari.

Jenis lampu yang nyata sederhana untuk mengontrol karena cahaya paralel tidak bervariasi, posisinya tidak masalah, hanya rotasi tidak, yang dikendalikan seperti *spot light* dengan memutar cahaya itu sendiri atau dengan memindahkan objek yang itu ditargetkan. Namun, jenis cahaya seharusnya tidak hanya terbatas pada sinar matahari.

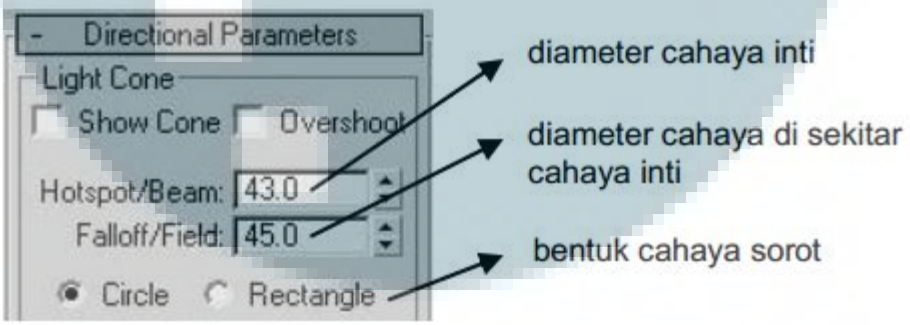
Lampu langsung juga sering digunakan untuk penerangan *fill*, yaitu pencahayaan sekunder yang melengkapi lampu utama skema. *Direct Light* adalah solusi yang baik terhadap cahaya ambient pemodelan, yang dapat dianggap sebagai *general light* tanpa arah, yang merupakan hasil dari cahaya yang tersebar dari permukaan model. Jenis cahaya yang paling terlihat dalam *scene* eksterior, ketika *Skylight* yang ini menghasilkan cahaya yang dipantulkan ke permukaan tidak terkena sinar matahari secara langsung.




Gambar 2.42. *Direct Lights*
(http://www.mahanani.web.id/2013/03/lightingpencahayaan-3d-max.html)



Gambar 2.43. *Direct Lights*.
(Mastering Autodesk 3ds Max 2013, hal 79)



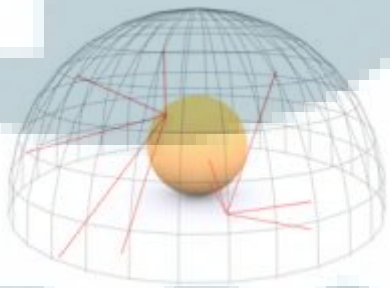

Gambar 2.44. Parameter *Direct Lights*
(http://www.mahanani.web.id/2013/03/lightingpencahayaan-3d-max.html)

d. *Skylight*

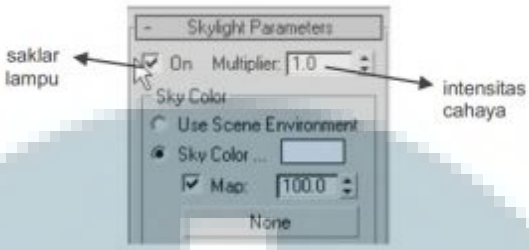
Jenis cahaya yang lebih baik adalah jenis *light Skylight*, yang bertindak sebagai kubah di atas *daylight* adegan dan model, yang dapat dianggap sebagai cahaya yang menyebar melalui atmosfer. Cahaya ini dapat digunakan dengan *renderer scan line*, dimana *renderer* tersebut mampu membuat bayangan yang halus. *Tracer Light* ini dirancang untuk digunakan dalam

adegan eksterior, dimana ia menghasilkan warna yang berhubungan dengan *Global Illumination*, seperti *radiocity*, modus ini juga sedikit lama dibandingkan dengan *mental ray*. Ini bisa menjadi solusi sangat cocok untuk adegan di luar ruangan, di mana komponen *daylight* dapat dengan cepat diatur tanpa tuntutan *radiocity*.

Gambar 2.45. Gambar *Skylight* 1
(http://docs.autodesk.com/3DSMAX/16/ENU/3ds-Max-Help/index.html?url=files/GUID-772D28D6-41C3-4C80-80AD-3774726E208F.htm,topicNumber=d30e441558)



Gambar 2.46. Gambar Area Cahaya *Skylight* 2
(http://docs.autodesk.com/3DSMAX/16/ENU/3ds-Max-Help/index.html?url=files/GUID-772D28D6-41C3-4C80-80AD-3774726E208F.htm,topicNumber=d30e441558)



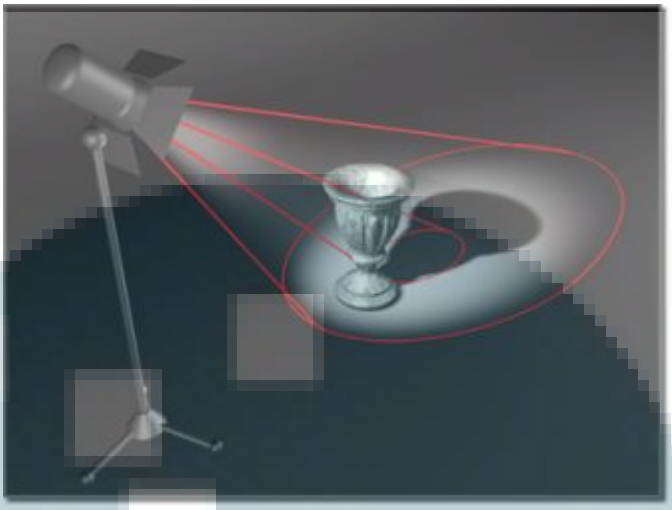

Gambar 2.47. Parameter di *Skylight*
(http://www.mahanani.web.id/2013/03/lightingpencahayaan-3d-max.html)

e. *Area Lights*

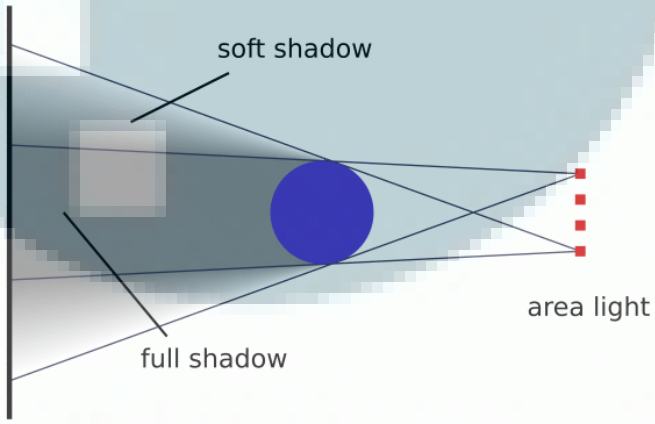
Seperti yang telah diketahui, di dunia nyata tidak ada sumber cahaya yang berasal dari satu titik (pencahayaan berasal dari sumber cahaya yang sangat kecil). Sumber cahaya kehidupan nyata selalu memiliki ukuran fisik. Lampu daerah menyediakan jenis cahaya dengan ukuran (atau bahkan *volume*), dari seluruh cahaya yang dipancarkan, menyediakan solusi yang jauh lebih realistis untuk menggambarkan *fiting* lampu sehari-hari.

Bila daerah lampu Anda lebih luas, bayangan akan menjadi lebih luas dan lembut, cahaya akan mengelilingi benda yang lebih besar dari karena ukuran sumber. Sebaliknya, jika cahaya yang sama dibuat semakin kecil, bayangan yang akan mempunyai ujung-ujung yang kasar. Pada titik ini area cahaya akan diperkecil ke ukuran yang cukup kecil untuk mulai bertindak sebagai titik cahaya.

*Area Lights* mampu membuat lampu yang sangat realistis, tapi kekurangannya adalah bahwa cukup menghabiskan banyak waktu untuk membuat lampu yang realistis.



Gambar 2.48. *Area Lights* 1
(http://docs.autodesk.com/3DSMAX/15/ENU/3ds-Max-Help/index.html?url=files/GUID-CA4D9F98-AB97-4782-8CA7-D90AD1770D56.htm,topicNumber=d30e573081)

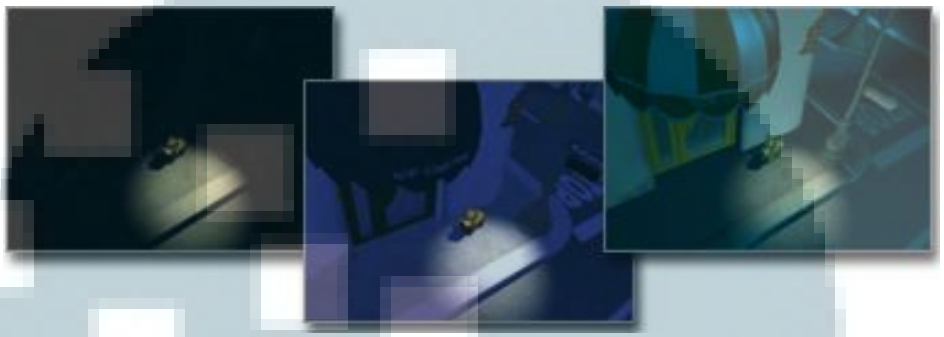



Gambar 2.49. *Area Lights* 2
(http://docs.autodesk.com/3DSMAX/16/ENU/3ds-Max-Help/index.html?url=files/GUID-772D28D6-41C3-4C80-80AD-3774726E208F.htm,topicNumber=d30e441558)

f. *Ambient Lights*

Jenis cahaya berikutnya adalah cahaya dapat ditempatkan dalam sebuah adegan, tapi cahaya yang dimaksud dalam *direct light section* seperti cahaya *ambient*. Pencahayaan menerangi seluruh adegan merata, sedangkan pada dunia nyata *ambient light* adalah penerangan umum yang biasa dialami dari

pantulan cahaya berbagai elemen di lingkungan. Untuk hasil yang realistis, *ambient light tools* lebih baik dimatikan, karena bisa membuat hasil yang tidak realistis pada *scene*.



Gambar 2.50. *No Ambient Lights(left), Low Ambient Lights (Center), User Defined Ambient Lights(right).*
(http://docs.autodesk.com/3DSMAX/15/ENU/3ds-Max-Help/index.html?url=files/GUID-1E2F3E07-4FD7-4DC5-B44F-0E4F917EB97F.htm,topicNumber=d30e599473)

### 2.5.2. *Photometric Light*

*Photometry* adalah pengukuran sifat dari cahaya. Dengan menggunakan *Photometric Light* dapat didapatkan hasil yang sangat realistis bahkan secara fisik mendetail dan akurat. Namun, yang harus diperhatikan adalah saat me*render* menggunakan cahaya *Photometric* akan nampak sangat realistis apabila *scene* diatur menggunakan *unit* yang realistis pula. Misalnya, sebuah bohlam lampu yang berada 1 meter akan sangat berbeda dengan sebuah lampu yang sama namun pada jarak 100 meter.

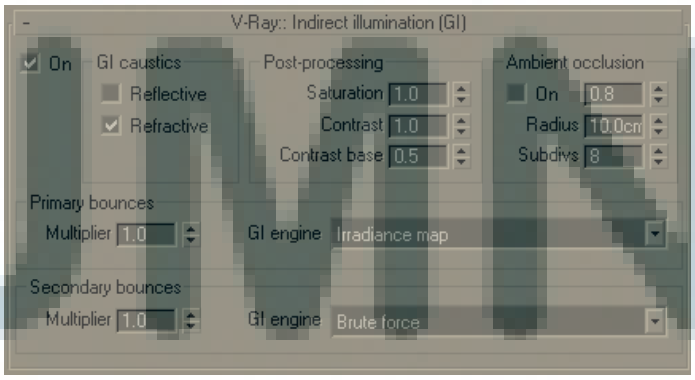
*Photometric Light* di dalam *3Ds Max* menggunakan beberapa jenis penyebaran cahaya yang berbeda, seperti: *Photometric Web File, Spotlight, Uniform Hemispherical* dan *Uniform Spherical*. Cara-cara penyebaran ini menentukan bagaimana cahaya dipancarkan dari sumber cahayanya. Cara

terakhir, *Uniform Spherical* (dalam *3Ds Max* 2008 namanya *Isotropic*) adalah cara termudah dan hanya bisa diterapkan pada 1 jenis lampu yaitu *Point Light*

Meskipun hanya ada dua jenis *light* fotometrik, *target* dan *free* (dan perbedaan antara kedua jenis ini hanya bagaimana kedua jenis *light* tersebut ditargetkan) hampir berbagai jenis lampu standar mungkin dilakukan dengan lampu ini karena model penyebaran cahaya yang berbeda. Selain itu, sementara pilihan distribusi mempengaruhi bagaimana cahaya tersebar di seluruh *scene*, bentuk cahaya yang digunakan untuk melemparkan bayangan ada enam pilihan: titik, garis, persegi panjang, *disc*, bola, silinder.

*Indirect Illumination (GI)*

*GI Caustic* merupakan cahaya yang telah melewati satu *diffuse*, dan satu atau beberapa *specular reflections* (atau *refractions*). *GI Caustic* dapat dihasilkan oleh *skylight*, atau benda *self-illuminated*. Namun, caustic disebabkan oleh *direct lights* tidak dapat disimulasikan dengan cara ini. Anda harus menggunakan bagian *Caustics* secara terpisah untuk mengontrol *Caustics direct lights.*




Gambar 2.51. Parameter yang Ada pada V-Ray
(http://www.mahanani.web.id/2013/03/lightingpencahayaan-3d-max.html)

*Refractive GI caustics* memungkinkan pencahayaan tidak langsung melewati objek transparan (misalnya kaca). Perhatikan bahwa ini tidak sama dengan *Caustics*, yang merupakan cahaya langsung menembus melalui benda-benda transparan.

Selain itu, *Reflective GI caustics* ini memungkinkan *indirect lights* dapat tercermin dari benda-benda yang memantul, seperti cermin. Berbeda dengan *Caustics*, yang merupakan cahaya langsung melalui permukaan yang memantulkan cahaya. Hal ini tidak aktif secara *default*, karena *Reflective GI caustics* biasanya berkontribusi sedikit untuk *final illumination*, namun *Reflective GI caustics* sering menghasilkan *subtle noise* yang tidak diinginkan.

### 2.7.2. *Three Point Lighting*

Tiga titik pencahayaan adalah sistem yang dibangun didalam sinematografi, dan ini adalah dasar yang sangat penting didalam pencahayaan di dalam CG. Salah satu alasan utamanya adalah teknik ini membantu menampilkan kesan 3 dimensi dari sebuah *scene* ataupun objek menggunakan lampu.

Sesuai dengan namanya, ada 3 lampu yang digunakan didalam teknik ini, dan masing- masing dari lampu mempunyai fungsi yang berbeda-beda. Yang membuat iluminasi utama dari sebuah *scene*, lampu utama adalah lampu yang paling mendominasi atau yang paling membentuk *shadow* yang paling gelap.

Ini bisa direpresentasikan sebagai sebuah lampu praktikal yang memancarkan lampu di sebuah *shot* yang dilakukan pada saat malam hari di dalam sebuah ruangan, cahaya matahari di siang hari di *outdoor*, atau cahaya matahari yang menembus melalui kaca di sebuah ruangan pada saat siang hari. Ini merepresentasikan *lighting* yang mendominasi dari sebuah *scene* yang memberikan petunjuk mengenai lokasi dari sebuah sumber cahaya.

Fungsi dari *fills* adalah untuk menciptakan *indirect light* yang terjadi pada saat *direct light* memantul dari permukaan sebuah benda, membuat penerangan dari sebuah *scene*. Penerangan utama biasanya diletakkan pada bagian seberang dari objek didalam bayangan dan membantu mengurangi kepadatan dari bayangan tersebut. Yang terakhir adalah *backlights* yang memberi kesan kedalaman pada *scene* dengan membantu memisahkan objek dengan latar belakang. Seringkali disebut sebagai *rim lights*, yang bekerja dengan cara menerangi bagian belakang sebuah objek, dengan demikian mereka membuat cahaya halus yang menerangi sisi ujung atau tepi pada sebuah objek.

### 2.7.3. *Key-to-Fill Ratio*

Hubungan antara *key light* dan *fill light* merupakan bagian penting didalam skema pencahayaan, yang mempunyai efek yang sangat signifikan didalam membuat *mood* pada saat *rendering*. Pada dsasarnya, hal ini ditentukan oleh rasio dari instensitas dari *key light* kepada *fill light*. *Key-to-fill ratio* yang tinggi mempunyai perbandingan *fill light* yang kecil, menjadikannya bernuasa lebih gelap dan *moody*, dengan ketajaman warna yang menonjol. Berbeda dengan *Key-to-fill ratio* yang rendah mempunyai *fill light* yang banyak, yang menghasilkan cahaya yang

lebih ringan, dengan ketajaman warna yang lebih tipis dan cahaya yang lebih terang serta membuat *mood* yang lebih senang.

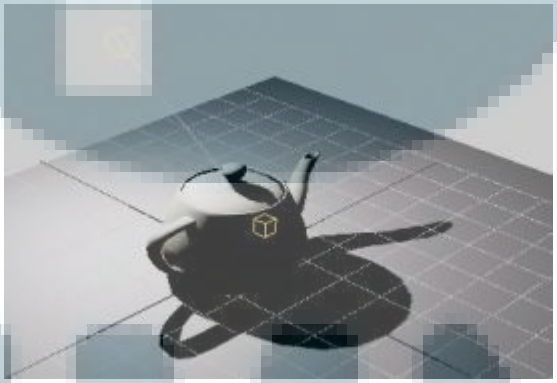
Gambar 2.52. *High Key-to-fill ratio*
(*Essential CG Lighting Techniques with 3ds Max*, hal 73)

Gambar 2.53. *Low Key-to-Fill Ratio*
(*Essential CG Lighting Techniques with 3ds Max*, hal 73)

### 2.7.4. *Shadow*

Bayangan memainkan peran besar dalam menggambarkan cahaya. Bayangan menambah, konsistensi realisme adegan ini, hubungan dan komposisi. Bayangan benar-benar menunjukkan hal-hal yang dinyatakan tidak mungkin untuk dilihat, serta merancang bayangan adalah suatu tugas signifikan sebagai merancang pencahayaan dalam sebuah adegan, begitu penting adalah peran mereka. Bahkan bisa digunakan untuk menutup bila terdapat kecacatan dalam *scene*.

Ketika menyiapkan skema pencahayaan di 3D, sejumlah besar waktu yang akan dihabiskan pada bayangan, karena banyak pilihaan untuk yang algoritma yang digunakan untuk menghasilkan baik keras dan lembut bayangan, dan bagaimana melakukan ini dengan cara yang seefisien mungkin.



Gambar 2.54. Contoh Bayangan Sebuah Benda
(http://docs.autodesk.com/3DSMAX/16/ENU/3ds-Max-Help/index.html?url=files/GUID-772D28D6-41C3-4C80-80AD-3774726E208F.htm,topicNumber=d30e441558)

## 2.8. *Rendering*

*Rendering* adalah proses dimana komputer menghasilkan gambar atau animasi yang detail dari suatu adegan, biasanya termasuk efek pencahayaan, pematerialan, serta refleksi. *Rendering* ditunjukan untuk memeriksa detail serta efek pada model

yang telah dibuat dan akan diserahkan pada tim produksi (Carver dan White, 2003).

Ketika memulai sebuah *render*, hal yang harus diperhatikan adalah tujuan dan fungsi dari gambar. Gambar yang dihasilkan tergantung dari *viewport* yang terlihat, biasanya tampilan *render* akan diambil dari segi perspektif, atau tampilan kamera yang telah di tentukan, oleh karena itu gambar tersebut akan mewakili proyeksi paralel. Hasil *render* dari segi perspektif lebih baik, namun lebih bagus lagi menggunakan kamera untuk menentukan hasil *render*.

Dalam *rendering*, ada tiga macam tipe *render* yang tersedia pada perangkat lunak *3Ds Max* (Wiradinata, 2008), yaitu:

1.) *Default Scanline Renderer*, *render* ini akan dipilih sebagai *render* awal pada *3Ds max*, *render* ini akan me*render* hasil dengan cara garis *horizontal*.

2.) *Mental Ray Renderer*, *render* ini menggunakan metodenya sendiri dalam *render* dan hanya dengan tipe *render* ini, berbagai macam material yang tersedia hanya dapat di*render* dengan menggunakan *render* ini.

3.) *VUE Filel Renderer*, *render* ini ditujukan untuk hasil yang lebih khusus yang dapat menghasilkan ASCII pada bagian yang di*render*.

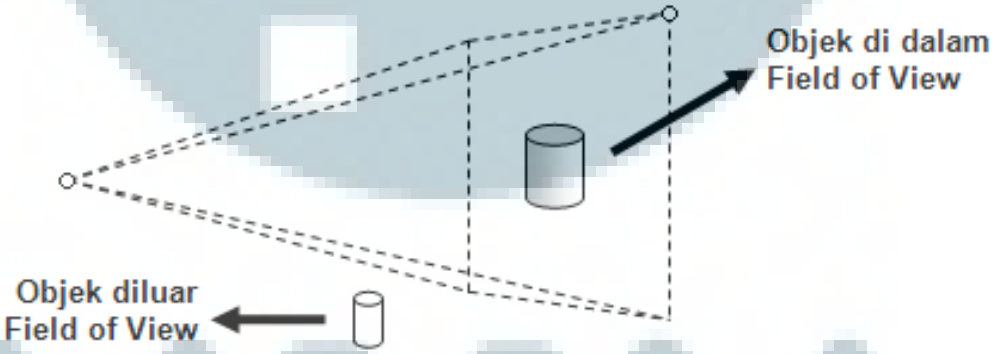
### 2.8.1. *Camera*

Dalam grafika 3D, sudut pandang *(point of view)* adalah bagian dari kamera. Kamera dalam grafika 3D biasanya tidak didefinisikan secara fisik, namun hanya

untuk menentukan sudut pandang kita pada sebuah *world*, sehingga sering disebut *virtual camera.*

Sebuah kamera dipengaruhi oleh dua faktor penting:

1. Faktor pertama adalah lokasi *(camera location).* Lokasi sebuah kamera ditentukan dengan sebuah titik (x,y,z).
2. Faktor kedua adalah arah pandang kamera. Arah pandang kamera ditunjukkan dengan sebuah sistem yang disebut sistem (U,N,V). Arah pandang kamera sangat penting dalam membuat sebuah citra, karena letak dan arah pandang kamera menentukan apa yang terlihat oleh sebuah kamera. Penentuan apa yang dilihat oleh kamera biasanya ditentukan dengan sebuah titik (x,y,z) yang disebut *camera interest.*




Gambar 2.55. Daya Tangkap Kamera.
(*Poly-Modeling with 3ds Max Thinking Outside of the Box*, hal. 106)

Pada kamera, dikenal *field of view* yaitu daerah yang terlihat oleh sebuah kamera. *Field of view* pada grafika 3D berbentuk piramid, karena layar *monitor* sebuah komputer berbentuk segi empat. Objek-objek yang berada dalam *field of*

*view* ini akan terlihat dari layar *monitor*, sedang objek-objek yang berada di luar *field of view* ini tidak terlihat pada layar *monitor*.

*Field of view* ini sangat penting dalam pemilihan objek yang akan diproses dalam rendering. Objek-objek diluar *field of view* biasanya tidak akan diperhitungkan, sehingga perhitungan dalam proses *rendering*, tidak perlu dilakukan pada seluruh objek.

Setelah menentukan posisi kamera, hal yang harus di perhatikan berikutnya menurut Carver dan White adalah menentukan *Focal Length*, *Focal Length* digunakan untuk menentukan seberapa lebar tampilan yang akan di*render*, semakin rendah *focal length* maka hasil yang ditampilkan semakin lebar, namun jika terlalu lebar membuat tampilan, maka akan menimbulkan distorsi pada gambar atau tidak alami.

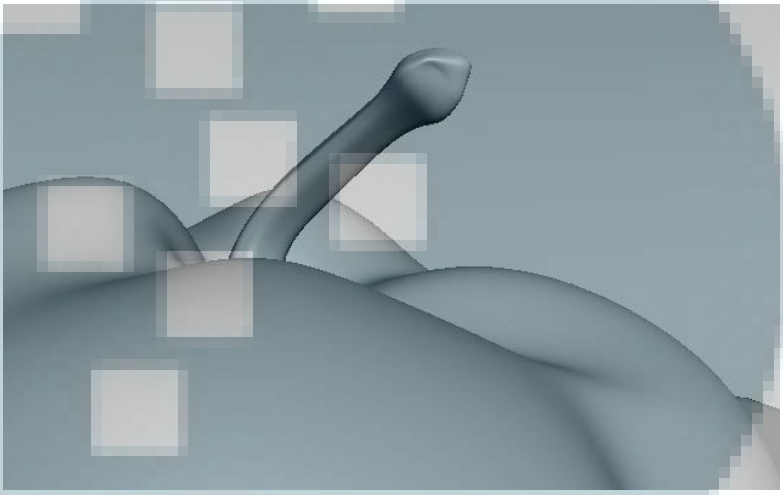
Membuat gambar dengan lensa 85mm atau lebih akan menghasilkan tampilan *close-up* objek dari jarak yang lumayan jauh dan tidak ada efek distorsi karena *focal length* yang meminimalisir hasil menjadi distorsi. Berbeda dengan menggunakan lensa 50mm, lensa ini akan menghasilkan tampilan yang mendekati dengan manusia melihat pada suatu objek, yaitu sekitar 40°. Jika menggunakan sudut sekitar 55° atau sama dengan menggunakan lensa 35mm, maka hasil tampilan akan terlihat senyata dan menyerupai benda aslinya.

### 2.8.2. Pengaturan *Rendering*

Pengaturan *rendering* merupakan hal yang harus dilakukan ketika akan me*render*, pengaturan ini merupakan ini dari hasil *render* yang akan terlihat (Derakhshani, Munn, McFarland, 2007), hal yang perlu di perhatikan adalah:

*Antialiasing*

*Aliasing* adalah efek tangga yang berada pada pinggir suatu garis atau warna, khususnya ketika objek tersebut melengkung (Gambar 2.56). Dengan *Antialiasing*, efek tangga akan akan diperhalus semaksimal mungkin namun memerlukan tambahan waktu dalam me*render*nya (Gambar 2.57).



Gambar 2.56. *Aliasing*
(Derakhshani, Munn, McFarland, 2007)

Gambar 2.57. *Antialiasing*
(Derakhshani, Munn, McFarland, 2007)

### 2.8.3. Elemen *Render*

Bayangan dan refleksi merupakan elemen penting dalam *render*, ketika elemen ini dipisah, maka *compositing* akan menjadi lebih bebas dikarenakan hasil dapat dirubah sebebas mungkin oleh mengatur bayangan dan refleksi. Ada tiga langkah yang harus diperhatikan untuk membuat hasil maksimal:

1. Bukalah *Render Elements* di dalam *Rendering Scene*, setelah itu pilih '*add'* untuk me*render* elemen yang dipilih, elemen yang dipilih sesuai urutan adalah:
    a. ***Alpha***, akan me*render* warna hitam dan putih, ini akan sangat berguna pada saat *compositing*.
    b. ***Reflection,*** akan me*render* hanya hasil refleksi dari objek.
    c. ***Refrection,*** akan me*render* objek yang trasparan.
    d. ***Self-Illumination,*** akan memisahkan *material* dari objek, sehingga intensitas dapat dimainkan pada saat *compositing*.
    e. ***Shadow,*** akan me*render* bayangan dari objek saja.
    f. ***Specular,*** akan me*render highlight* dari material objek.
    g. ***Z-Depth,*** akan me*render* dengan abu-abu, namun semakin terang warna abu-abu tersebut maka objek tersebut dekat dengan kamera, sebaliknya, semakin gelap warna abu-abu tersebut maka objek semakin jauh dengan kamera.
2. Setelah memilih elemen *render*, elemen tersebut akan masuk kedalam daftar *Element Rendering,* kemudian pada bagian *Selected Element*

*Parameters*, pilih tempat dimana akan menyimpan hasil *render* tersebut dalam bentuk *bitmap*.

3. Jika ingin setiap elemen ter-*render*, maka centanglah *Display Elements*.
4. Mulailah *render*, maka 3ds max akan me*render* dan menyimpan elemen yang telah dipilih.