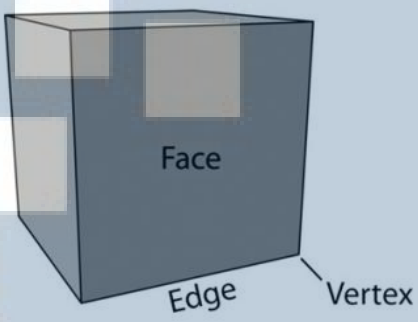
# BAB II

# TINJAUAN PUSTAKA

## 2.1. *3D Image*




Gambar 2.1. *3d Image*
(Copine,2011)

Menurut Copine (2011) pada bukunya yang berjudul *3D Art Essentials The Fundamentals*, 3D object adalah sebuah gambaran visual yang tersusun dari susunan *face* atau yang biasa disebut *polygon* yang terbetuk dari garis disetiap sisinya atau dikenal dengan istilah *edge* yang dibentuk dari titik titik kecil yang disebut *vertex.* Jadi misalkan ada sebuah kubus dalam 3D dapat di ibaratkan terbentuk dari 6 buah *polygon* yang setiap buah nya terbentuk dari 4 *edge* dan 4 *vertex.*

## 2.2. *Workflow*

Keathley (2014) pada bukunya yang berjudul *Digital Asset Management* menyatakan bahwa, *Workflow* adalah usaha dari mesin dan manusia untuk melakukan suatu rangkaian kerja yang terstruktur. Perangkat lunak perlu menyesuaikan dan merespons variabel dari apa yang manusia atur. Pengguna

perlu memahami praktik terbaik yang berkaitan dengan alur kerja, dan keduanya membutuhkan peranan masing-masing untuk menghasilkan dan mendistribusikan suatu tugas.

## 2.3. *Pipeline*

*Pipeline* dari animasi adalah sekumpulan orang, *hardware* dan *software* yang bekerja dalam urutan tertentu dalam pembuatan animasi atau asset 3 dimensi. Pada sebuah *pipeline*, tahapan produksi dibagi menjadi beberapa tahapan besar yang dapat dirinci dalam beberapa tahapan kecil. Tahapan kecil tersebut dibagi kepada beberapa orang untuk dikerjakan bersama sehingga pekerjaan animasi atau asset 3 dimensi lebih efisien (Beane, 2012, hlm. 21-22).

Renee Dunlop (2014) menyatakan pula bahwa *pipeline* juga merupakan tindakan antisipasi jika terjadinya suatu hambatan pada proses produksi suatu proyek. Jika terjadi suatu masalah ditengah produksi maka dalam *pipeline* telah diberikan jawaban kemana jalur selanjutnya jika terjadi masalah disuatu tahapan produksi. Pada *flowchart* produksi telah diberikan penjelasan tentang kemana saja jalur yang diambil ketika suatu data telah dikerjakan. Beberapa pipeline juga membagi pekerjaan menjadi 2 tahap, misal *raw modelling* yang akan diperiksa oleh *director* kemudian asset tersebut akan dikerjakan detailnya oleh artist lain sehingga tidak ada pekerjaan yang tertunda pada satu artist.

Menurut Pellacini (2009) beberapa tahapan atau divisi pada suatu *pipeline*, yaitu: *Story Development, Editorial Development, Art Development, Casting, Modeling, Shading /texturing, Rigging, Set Dressing, Layout, Animation, Simulation, Effect, Lighting, Rendering.*

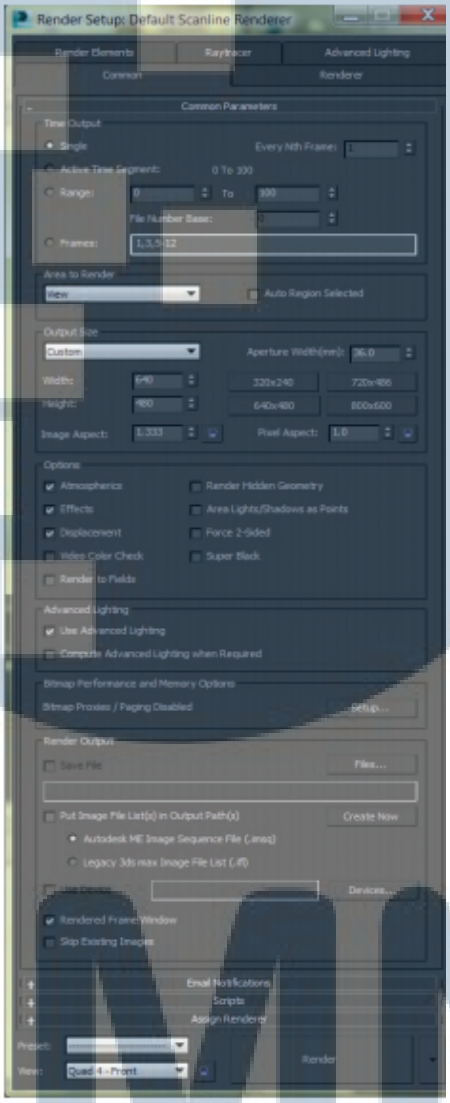
## 2.4. *Automation Theory*

*Automation*/Otomasi adalah penggunaan *system control* yang digunakan pada suatu *platform* yang sengaja diprogram untuk berjalan sendiri. Otomasi dibuat untuk mengendalikan suatu mesin tanpa intervensi manusia (Rodic, 2009). Ciri-ciri dari suatu proses yang dapat diotomasi adalah suatu rangakaian kerja yang terdiri dari urutan kerja/ hirarki. Dari hirarki tersebut beberapa pekerjaan dapat dikerjakan secara bersamaan dan dapat diselesaikan dengan sekali proses. Kemudian pekerjaan yang melalui sistem yang dilakukan secara berulang-ulang. Namun pada *automation* tidak dapat mengecek kualitas suatu barang produksi yang bersifat ekslusif. Perlu adanya intervensi manusia untuk mencapai kualitas tertentu.

## 2.5. *Time Efficiency*

Menurut König (2007) dalam jurnalnya yang berjudul *The Journal Of Psychology*, menjelaskan bahwa tujuan utama dari suatu usaha adalah untuk menghasilkan suatu hasil yang efektif dan produktif dengan *time management*. Jadi untuk mendapat hasil yang sesuai diusahakan juga menggunakan waktu yang singkat. Membutuhkan strategi dimana manusia dapat mengelola waktu, uang, energi dan sumberdaya manusia untuk mencapai hasil yang sesuai dengan keinginan dan mencapai target.

## 2.6. *Render*




Gambar 2.2. *The Common tab in the Render Setup dialog box*

(Autodesk 3ds Max 2014 Essentials, 2013)

*Rendering* adalah langkah terakhir dalam membuat karya CG Anda, namun ini adalah langkah pertama untuk dipertimbangkan saat Anda mulai membuat adegan. Selama *rendering*, komputer menghitung sifat permukaan pemandangan, pencahayaan, bayangan, dan pergerakan objek dan kemudian menyimpan urutan

gambar. Untuk sampai ke titik di mana komputer mengambil alih, *user* harus menyiapkan kamera dan membuat pengaturan sehingga Anda akan mendapatkan apa yang Anda butuhkan dari tempat kejadian (Derakhshani, 2013 hlmn. 331).

### 2.6.1. *Pre-Render/Render Setup*

*Render setup* adalah langkah dimana user mengatur parameter pada render setup dialog yang berisi tentang pengaturan jumlah frame yang di-*render,* besar resolusi gambar hasil *render,* dan banyak pengaturan lainnya yang membantu *user* untuk menentukan hasil akhir visual dan wilayah penyimpanan *file*. Setiap file harus melalui tahapan ini agar *asset* yang di proses dapat mencapai ekspektasi dan rancangan visual.

### 2.6.2. Metode Dasar *Rendering*

Ada beberapa cara dalam menghasilkan atau melakukan render suatu gambar dari suatu objek 3D. Render engine adalah sistem pada 3DS Max yang digunakan untuk menghasilkan gambar digital yang telah memiliki data alogaritma yang unik. Setiap *render engine* memiliki karakteristiknya masing-masing.

#### 2.1.1.1. Scanline

Alogaritma *render* pada *scanline* sangat cepat, kelemahan dalam sistem render ini adalah tidak bisa menghitung reflection, refraction, atau *global illumination* yang kompleks seperti yang ada sekarang ini. Seperti yang dapat dilakukan oleh menta-Ray, V-Ray, atau Renderman. *Renderer* ini sangat baik digunakan pada saat *pre-render visual*. Metode ini cocok untuk mendapatkan visual look yang *Cartoonlike, flat,* dan *shell-shaded*.

#### 2.1.1.2. Raytracing

Metode *render* melalui *raytracing* lebih baik dari pada *scanline.* Raytrace dapat mengkalkulasikan *reflection, refraction,* dan banyak pengaturan lainnya dalam pilihan *render*. Cara kerjanya dengan menimbulkan cahaya berupa bentuk benda yang berinteraksi atau berdekatan pada objek. Pada titik sample jika permukaan tersebut bersifat reflektif maka akan diberi informasi seberapa jauh cahaya itu akan dipantulkan dan apakah akan memantul kembali ke objek lain yang non-reflektif. Raytracing lebih mendekati realis dibanding dengan *scanline*.

### *2.6.3. Global Illumination*

*Global Illumination* adalah istilah umum untuk mendeskripsikan sekelompok algoritma dan metode untuk menciptakan suatu tata pencahayaan yang realistis dan *shader* pada *render engine* pada saat *rendering*. Metode ini biasanya bertumpuan pada raytracing namun akan ditambah beberapa fungsi agar hasilnya dapat semakin realis.

#### 2.1.1.3. Photon Mapping

Algoritma *render* dari *global illumination* yang mengikuti metode dari raytracing, tapi digunakan secara terbalik. Cara tersebut adalah *Photons* dipancarkan dari sumber cahaya dan memantul di sekitar *scene* tersebut. Photon meninggalkan titik/berkas cahaya.

#### 2.1.1.4. Image-Based Lighting

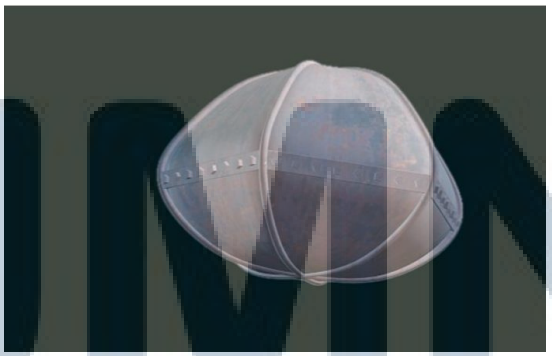
Cara ini merupakan cara dimana user membuat *surrounding dome* atau *sphere* yang memberikan cahaya terhadap scene berdasarkan gambar yang terdapat pada

*dome* atau *surround sphere* tersebut. *Dome* atau *surrounding sphere* biasanya diisi oleh gambar pemandangan.

### *2.6.4. Multi Passes Render*

Birn (2014) pada buku yang berjudul *Digital Lighting and Rendering* menjelaskan bahwa *multi pass render* adalah sebuah hasil gambar yang di-*render* yang terdiri dari beberapa lapisan yang disebut *pass* atau *element.* Lapisan ini mengandung beberapa informasi yang menyusun suatu perhitungan gambar yang dibutuhkan dalam membentuk hasil akhir render. Lapisan ini memudahkan kita dalam mengulang hasil pengaturan render seperti *lighting*, *shadow*, bahkan *Z-depth*. Biasanya pengaturan pada render pass dapat dilakukan pada proses pasca produksi berikut adalah macam-macam pass pada render pass.

#### 2.1.1.5. Diffuse Pass

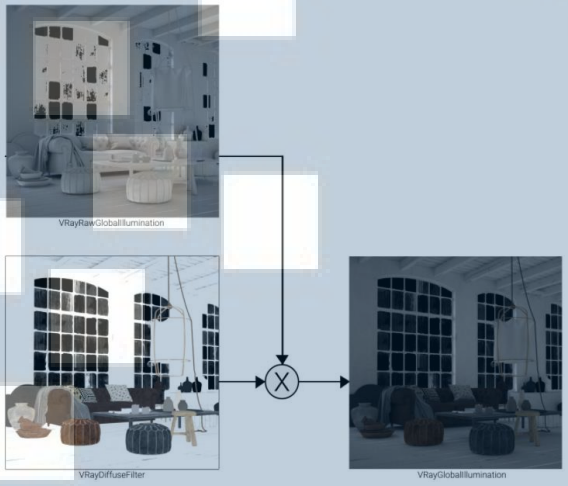


Gambar 2.3. *Difuse Pass*

(Jeremy Birn, 2014)

Merupakan *pass* yang mengandung warna penuh dari subyek yang akan kita *render*. *Pass* ini tidak mengandung *reflection* atau *highlight* karena *diffuse pass* mengandung *diffuse illumination* dari cahaya. Permukaan telah memiliki shading terang pada tempat cahaya yang datang, begitu

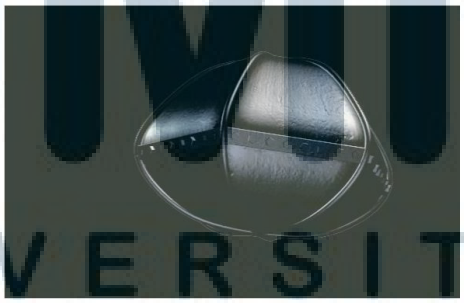
juga sebaliknya pada tempat yang sedikit terkena cahaya. *Pass* ini juga mengandung texture yang telah di-*assign.*




Gambar 2.4. *VrayGlobalilumination*
(https://docs.chaosgroup.com/display/VRAY3MAX/Global+Illumination+Render+Elements, 2017)

Pada situs docs.chaosgrup.com menjelaskan bahwa pada V-Ray *render engine* dapat memakai *pass* serupa yaitu VRayGlobalIlluminati**o**n yang dihasilkan dari VRayRawGlobalIllumination x VRayDiffuseFilter.

#### 2.1.1.6. *Specular Pass*



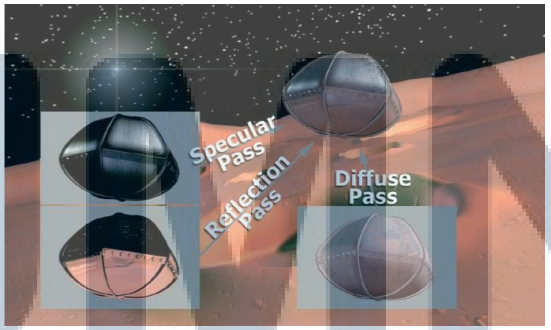
Gambar 2.5. *Difuse Pass*
(Jeremy Birn, 2014)

Specular pass (highlight pass) adalah *pass* yang mengandung isolasi dari highlight yang ada pada objek render. *Diffuse shading* dan *color mapping* akan dibuat menjadi warna hitam.

Gambar 2.6. *VRaySpecular*
(https://docs.chaosgroup.com/display/VRAY3MAX/Specular+%7C+VRaySpecular, 2017)

Pada situs docs.chaosgrup.com menjelaskan bahwa pada V-Ray *render engine* dapat memakai *pass* serupa dengan nama VRaySpecular.

#### 2.1.1.7. *Reflection Pass*




Gambar 2.7. *Reflection Pass*
(Jeremy Birn, 2014)

*Reflection pass* adalah *pass* yang mengandung informasi pantulan bayangan dari bayangan objek, pantulan dari objek lain atau pantulan dari *environment* pada objek 3D.

Gambar 2.8. *VRayReflection*
(https://docs.chaosgroup.com/display/VRAY3MAX/Reflection+%7C+VRayReflection, 2017)

Pada situs docs.chaosgrup.com menjelaskan bahwa pada V-Ray *render engine* dapat memakai *pass* serupa dengan nama VRayReflection.

#### 2.1.1.8. *Ambient Pass*

Gambar 2.9. *Ambient Pass*
(Jeremy Birn, 2014)

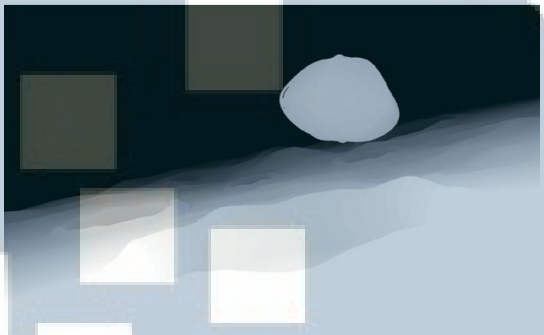
*Ambient Passes* (*Color Pass)* dapat disebut juga warna dan *texture map* pada objek. *Pass* ini membuat seolah permukaan objek mendapat pencahayaan dari lighting sekitar dan terlihat bentuk dasar dari benda tersebut.

#### 2.1.1.9. *Occlusion Pass*

Gambar 2.10. *Occlusion Pass*
(Jeremy Birn, 2014)

Pada VRay *render engine, oclussion pass* dapat dihasilkan melaui VRayDirt. VRayDirt merupakan *texture map* yang menstimulasi adanya bercak-bercak yang biasanya ada pada celah objek. VRayExtraTex adalah *pass* yang dipakai untuk menghasilkan data *render pass image.* Hal ini dikarenakan VRayDirt hanya membantu mengatur besar kecilnya pengaruh *dirt texture* namun tidak menghasilkan *render pass image.* Maka jika keseluruhan *occlusion pass* ingin didapat user harus menambahkan VRayExtraTex pada *render element*.

#### 2.1.1.10. *Depth Pass*



Gambar 2.11. *Depth Pass*
(Jeremy Birn, 2014)

*Depth pass* (Z-depth atau Depth Map) merupakan *pass* yang menyimpan informasi akan kedalaman pada setiap objek terhadap kamera. *Pass* ini berisi serangkaian nilai yang mengukur jarak dari kamera kepada objek.

Kegunaan pass ini:

a. Memanipulasi *fog*, *haze*, dan atau atmosfer pada tahap *compostiting*
b. Memanipulasi menjadi tumpuan pada pemberian efek *Depth of Field* dan *bokeh*
c. User dapat melakukan *render* pada banyak *layer*, dengan *depth pass* di setiap layernya. Hal ini akan membantu objek dari *layer* lain untuk menemukan *space* yang tepat pada *scene* tersebut

### 2.6.5. *Basic Render Workflow*

Biasanya seorang *lighting artist* juga merupakan orang yang mendapat tugas pada render pula. Berikut adalah *basic rendering workflow* (Beanne, 2012, hlm 244-245)

a. Menentukan *setting* dari *lighting*. *Artist* harus yakin *lighting* telah menerangi *scene*.
b. Artist mengkaji apakah *scene* tersebut membutuhkan *advance lighting render* seperti *global illumination* atau *image-based lighting*. Jika iya, maka *artist* akan mengatur *render* untuk mengevaluasi pengaturan *lighting* tersebut.
c. Menentukan pengaturan *render*. *Artist* menentukan *render engine* untuk dilanjutkan menuju tahap *rendering* tingkat selanjutnya. Karena setiap *render engine* memiliki standar yang berbeda dalam menghaslikan gambar akhir.
d. *Artist* menentukan *render passes*. *Artist* menentukan *render passes* yang dibutuhkan.
e. *Render image*. Artis melakukan render untuk mendapatkan *image sequence.*
f. Proses *Compositing render pass* untuk membangun gambar final/*final look.*

## 2.7. VRAY

Kuhlo dan Eggert (2010) dalam bukunya *Achitectural Rendering with 3dMax and V-ray* menjelaskan bahwa ada beberapa alasan mengapa menggunakan VRay sebagai *render engine*. VRay merupakan contoh dari jenis raytracer yang melakukan kalkulasi cahaya menggunakan pantulan benda yang reflektif untuk mencapai pencahayaan yang realistik. VRay merupakan *plug-in* render yang tersedia pada banyak software 3D. Pengaturannnya pada setiap program pun cenderung sama. VRay *renderer* juga sering dipakai pada produksi film dan periklakan. *Renderer* ini juga memiliki banyak komunitas yang membantu perkembangan dari *renderer* ini. Komunitas ini sering kali membuat penemuan atau sebuah metode pengaturan untuk menghasilkan suatu tampilan visual tertentu. Berikut adalah contoh dari beberapa render element yang ada pada VRay *render engine.*

a. VRayGlobalIllummination

   VRayGlobalIllummination menyimpan informasi tentang pencahayaan tidak langsung. Yang dimaksud dengan pencahayaan tidak langsung adalah cahaya yang dibentuk dari pantulan cahaya pada area sekitar benda.

b. VRayReflection

   VRayReflection menyimpan informasi mengenai kalkulasi nilai refleksi material benda yang ada di sekeliling benda.

c. VRayRefraction

   VRayRefraction berisi pembiasan cahaya sesuai dengan nilai yang diatur pada material.

d. VRaySpecular

VRaySpecular menampilkan pembiasan yang sesuai dengan *value* yang diatur pada material. Dapat digunakan untuk mengatur *brightness* tanpa harus melakukan *render* ulang.

e. VRayLighting

VRayLighting menyimpan pencahayaan langsung dari lampu yang ada di *scene* serta *illumination* yang menyebar.

f. VRayZDepth

VRayZDepth merupakan gambar *grayscale* yang menyimpan informasi mengenai jarak setiap benda terhadap kamera. Benda paling dekat dengan kamera akan berwarna hitam sedangkan benda yang lebih jauh akan berwarna abu-abu. Semakin terang warna abu-abu maka semakin jauh benda tersebut.

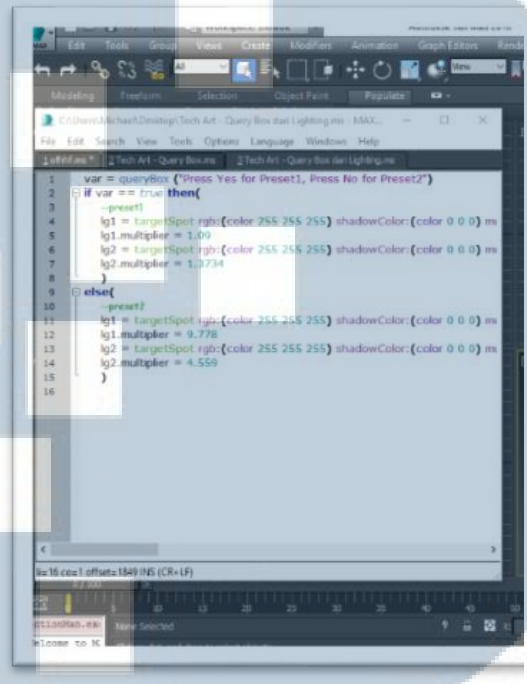
g. VRaySSS2

VRaySSS2 merupakan *SubSurface Scattering* yang dapat membantu pewarnaan material yang sedikit diterobos cahaya pada saat terkena cahaya. Pass ini dipakai biasanya untuk mengatur warna pada bagian kulit.

h. VrayExtraTex

VRayExtraTex membuat pemetaan *texture* pada suatu scene di-*render* menggunakan suatu *texture map*. *Texture* yang diaplikasikan dapat berupa *bitmap* atau *texture map* seperti VRayDirt atau VRayCurvature.

## 2.8. 3dsmax MAXScript



Gambar 2.12. 3dsmax MAXScript

(Dokumentasi Penulis, 2017)

3dsmax MAXScript adalah suatu fasilitas di dalam software 3dsmax yang dapat membuat kita berkomunikasi secara interaktif kepada software 3dsmax. MAXScript dibuat sama dengan sangat identik dengan semua user perintah pada *user interface*. Jadi *user* dapat sangat bebas memberikan perintah melalui MAXScript (Lama, Johnson, Maffei, Bousquet, 2007, hlm.10). MAXscript mendukung pengerjaan otomatis secara berulang ulang seperti membuat obyek, *selecting*, menggerakan, *import*, mengatur *Pre-Render*, dll.

### 2.8.1. Logika MAXScript

Dalam bahasa pemrograman tentunya ada aturan yang digunakan dalam menulis script. Berikut ini adalah beberapa contoh logika atau aturan dalam menulis *script*.

a. *Comments*

```
b = box()
b.length = 20.0
-- Here is a comment.
-- Here is a second comment, which continues
-- on this line. The next line is part of the script.
b.width = 30.0
```

Gambar 2.13. *Comments in* 3dsmax MAXScript
(Lama, Johnson, Maffei, Bousquet, 2007)

*Comments*/keterangan dapat dilakukan dengan menggunakan (--). Cara ini digunakan untuk menandai bagian *script* agar rapi dan memudahkan *programmer* untuk memeriksa *script*.

b. *Multiline statement*

```
torus radius1:10 \
    pos:[0, 0, 20] \
    wirecolor:[225, 230, 100]
```

Gambar 2.14. *Multiline command in* 3dsmax MAXScript
(Lama, Johnson, Maffei, Bousquet, 2007)

*Statement* adalah jenis baris yang berisi perintah yang digunakan untuk berkomunikasi kepada 3dsmax. Baris ini bisa saja sangat panjang kesamping, namun *programmer* biasanya lebih sering menyusunnya kebawah agar terlihat rapi dan mudah dikoreksi jika terjadi *error*.

c. Variabel dan data

Di dalam kebanyakan Bahasa pemrograman beberapa *item/statement* akan diwakili oleh variabel agar lebih mudah dalam melakukan *coding*. Sedangkan tipe data yang digunakan dalam Maxscript adalah *number*(angka), *string*(huruf), dan Boolean (*true/flase*).

### 2.8.2. MAXScript *listener*

Merupakan fasilitas dalam 3dsmax yang gunanya adalah merekam semua *command* yang dilakukan pada *user interface*. Untuk membuka fasilitas tersebut adalah Scripting>MAXScript Listener atau bisa menggunakan F11. Semua *command* yang dilakukan bisa terekam pada fasilitas ini namun beberapa *command* tidak bisa terekam. Jadi jika penulis memiliki kebutuhan *command* yang tidak terekam penulis harus mencari dari *website* milik Autodesk 3dsmax *MAXScript command.*

## 2.9. *Command-Line Rendering*

*Windows Command-Line* adalah sebuah sistem operasi yang dibuat dalam *Microsoft Windows* dan diakses melalui *command shell-window*. Setiap versi *Windows* memiliki *built-in command-line,* yang dapat digunakan untuk menjalankan perintah *Built-in Commands, Utilities,* dan *Script.* Meski *Command-line* sangat berguna, beberapa user tidak memakainya.

*Windows Command-Line* dapat digunakan untuk menjalankan *batch program* atau *Windows script*. Jika Anda ingin menggunakan fasilitas ini Anda harus menyusun jalur file untuk *Command* line mencapai program tujuan. Serta memasukan *setting parameter*, agar jika program tujuan telah aktif pengaturan pada program telah siap digunakan (Stanek, 2015, hlm117).

Sebuah fasilitas yang membuat *user* dapat melakukan *batch render* atau render secara massal. Selain itu juga dapat mengubah *render parameter* seperti jumlah *output frame,* resolusi, mengubah dan lain sebagainya. Fasilitas ini dijalankan

oleh 3dsmaxcmd.exe yang berada di *installation folder* pada data 3dsmax. Fasilitas *command-line rendering* juga mendukung *network rendering* dan membiarkan *Backburner utility* yang mengatur pekerjaan dengan sistem yang saling terhubung.

### 2.9.1. *Batch Render*

*Command prompt* memiliki mode *batch* yang mendukung dalam menjalankan serangkaian perintah. Dalam *batch command* perintah dibaca dan dieksekusi satu persatu. Biasanya *batch command* dilakukan melalui *script file.* Namun *batch command* dapat juga diketik langsung dalam *command prompt* sama seperti melakukan *command* pada umumnya untuk memproses satu set file. Berikut adalah contohnya:

```
"C:\Program Files\Autodesk\3ds Max
2016\3dsmaxcmd.exe"camera:"PhysCamera001"start:"0"end:"0" "D:\CCTV jadi"

PAUSE
```

Setelah itu di-*save* dengan format (name).*BAT,* dan jika dijalankan akan muncul *command prompt.* Namun kekurangan dari fasilitas ini adalah tidak adanya pemberitahuan saat program dijalankan, jadi bisa saja dapat merusak dan memanipulasi file *user* tanpa pemberitahuan. Namun hal yang perlu diketahui adalah jika terdapat *costum* data pada *command-line rendering* maka pengaturan file mengikuti *data save* pada *render setup file* tersebut.