BAB II

TINJAUAN PUSTAKA

2.1. Animasi

William (2009) menyebutkan bahwa konsep animasi sudah ada sejak 35,000 tahun yang lalu. Manusia sudah membuat lukisan gua yang menggambarkan hewan dengan empat pasang kaki untuk mendapatkan ilusi gerakan. Sejak saat itu, manusia kerap menciptakan inovasi baru untuk menghidupkan sebuah gambar dan berhasil menemukan prinsip *'The Persistence of Vision'*.

Dengan pesatnya perkembangan teknologi, animasi telah berkembang ke arah animasi tiga dimensi dan menjadi bagian yang penting di dalam industri. Beane (2012) menjelaskan bahwa animasi tiga dimensi merupakan sebutan yang melingkupi keseluruhan industri yang menggunakan grafis tiga dimensi melalui *software* atau *hardware* komputer. Industri tersebut berkembang dan digunakan dalam dua bidang utama, yaitu hiburan dan penelitian ilmiah.

2.1.1. 12 Prinsip Dasar Animasi

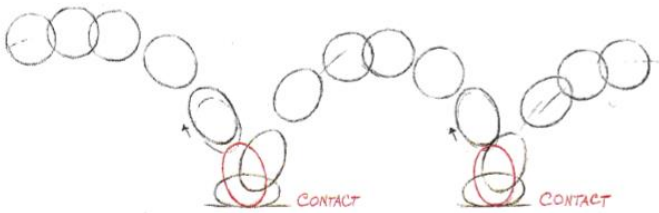
Pada 1981, Thomas & Johnston menetapkan dasar-dasar prinsip yang terdiri dari dua belas poin, yaitu:

- *Squash and Stretch*
- *Anticipation*
- *Staging*

- *Straight Ahead and Pose-to-Pose*
- *Follow Through and Overlapping Action*
- *Ease In, Ease Out*
- *Arcs*
- *Secondary Action*
- *Timing*
- *Exaggeration*
- *Solid Drawing*
- *Appeal*

Berdasarkan 12 prinsip yang telah disebutkan, penulis berfokus ke dalam prinsip *exaggeration, squash and stretch*, dan *arcs* yang dijelaskan secara lebih detail.

- ***Squash and Stretch***



Gambar 2.1. *Squash and Stretch* di dalam *Bouncing Ball.*

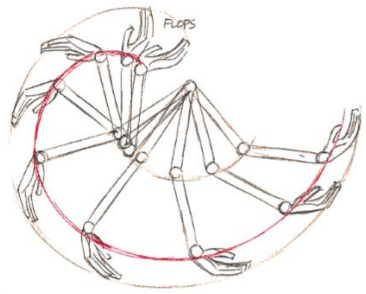
(Williams, R. (2009). *The animator's survival kit*: Expanded edition. Hlm. 94)

Perbedaan terbesar dari benda hidup dan mati adalah pergerakannya. Benda hidup, sekaku apapun selalu memiliki perubahan atau distorsi bentuk dalam gerakan yang dialaminya, tidak seperti benda mati yang kaku. Thomas dan Johnston menjelaskan gerakan ini melalui prinsip *squash dan stretch* (1981). *Squash* merupakan keadaan

sebuah benda yang ditekan oleh tekanan yang besar, sedangkan *Stretch* merupakan keadaan sebuah benda yang dibentangkan semaksimal mungkin.

Animator seringkali menggunakan prinsip tersebut dengan sangat ekstrim, menurut Thomas & Johnston. Oleh karena itu, beliau memberikan nasihat agar gerakan yang dirancang tidak terlihat aneh, yaitu dengan memperhatikan volume benda yang mengalami distorsi tersebut. Walaupun benda padat mengalami distorsi bentuk, volume benda bersifat tetap.

- ***Arcs***



Gambar 2.2. *Arcs* di dalam Gerakan Tangan

(Williams, R. (2009). The animator's survival kit: Expanded edition. Hlm. 90)

Semua makhluk hidup tidak dapat bergerak dalam bentuk garis lurus seperti mesin, melainkan selalu bergerak di dalam bentuk yang sirkular. Hal ini disebut sebagai *arcs* oleh Thomas dan Johnston (1981).

- ***Exaggeration***

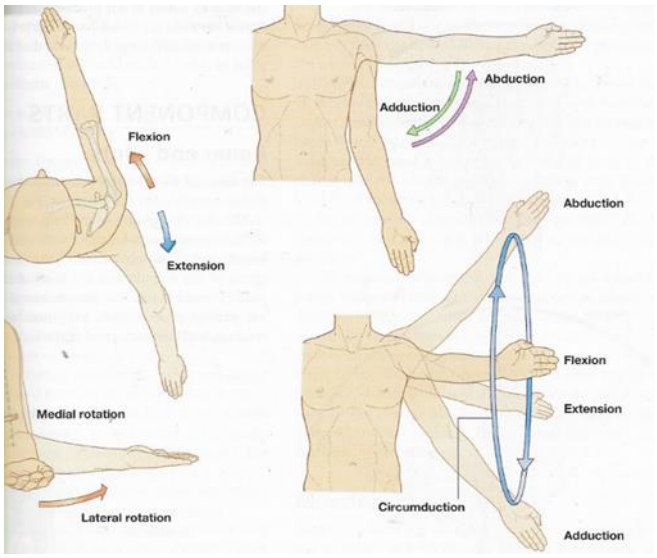
Exaggeration berarti melebih-lebihkan menurut terjemahan langsung Bahasa Indonesia. Walaupun begitu, Walt Disney (di dalam Thomas & Johnston, 1981) mengartikan kata ini sebagai ‘lebih dipercaya’ atau ‘lebih natural’. Menurut beliau, apabila seseorang sedih, buatlah dia menjadi sangat sedih. Dalam animasi, hal ini dapat dicapai dengan bentuk yang sangat ekstrim dan terdistorsi.

2.2. Anatomi Sendi Manusia

Jembatan dari dua elemen dari sistem tulang yang memungkinkan gerakan pada tubuh manusia merupakan sistem sendi. Drake et al. (2005) membagi sendi berdasarkan gerakan-gerakan dasar yang dihasilkan dari sendi tersebut:

- Rotasi (*Spin*) berdasarkan sebuah poros *axis*. Gerakan ini dapat mempengaruhi tulang dan otot. Gerakan otot dibagi menjadi gerakan melintir ke dalam dan gerakan mengendurkan (*Supinated*).

- Ayunan (*Swing*) menggerakan tulang ke sudut dan arah tertentu. Gerakan ini dibagi menjadi gerakan menekuk (*Flexion*) dan gerakan meluruskan atau meregang (*Extension*). Terdapat pula gerakan membuka (*abduction*) dan gerakan merapat atau menutup (*adduction*).



Gambar 2.3. Jenis Gerakan Manusia

(Drake et al., 2005, Gray's Anatomy for Students. hlm. 34)

2.2.1. Kelenturan (*Flexibility*)

Di dalam Kamus Besar Bahasa Indonesia, fleksibel berarti sesuatu yang lentur dan mudah dibengkokkan atau cepat menyesuaikan diri. Fleksibilitas merupakan batas maksimal dari kemampuan gerak sebuah sendi. Seseorang yang lentur mampu membungkuk dan merenggangkan tubuhnya dengan maksimal. Selain itu, Bompa (2000) juga menjelaskan fleksibilitas sebagai luasnya jangkauan gerakan dari sebuah sendi (dalam Mekayanti, 2015).

2.3. *Nodes* dan Hierarki

Menurut Palamar (2016), *nodes* merupakan kepingan-kepingan data yang membangun sebuah file Maya. Kepingan-kepingan tersebut memiliki peran dan fungsi masing-masing, dirakit menjadi sebuah tampilan 3D di dalam *software* Maya. Setiap langkah

yang dilalui oleh pengguna dicatat melalui setiap perubahan yang dialami oleh *nodes* yang bersangkutan. Selama sejarah *nodes* tercatat, pengguna dapat kembali merubah langkah yang telah dilalui. Selain itu, *nodes* dapat mencatat perubahan paling baru sebagai dasar awal *nodes* tersebut dengan menghapus sejarah yang ada.

Maya membagi data *nodes* atau atribut menjadi dua jenis, *transform node dan shape node* yang hubungannya diatur di dalam DAG (*Directed Acrylic Graph*). Palamar (2016) menjelaskan bahwa data yang mengatur letak, ukuran, dan sudut sebuah benda dicatat di dalam *transform node* dan data bentuk dari sebuah benda terletak di dalam *shape node*. Pengaturan DAG meletakkan node-node tersebut di dalam sebuah hierarki dimana *shape node* merupakan anak dari *transform node*. Anak di dalam sebuah hierarki terletak di dalam dan di bawah orang tuanya. Setiap perubahan yang dialami orang tua selalu diikuti oleh anaknya.

2.3.1. Attribute

Atribut merupakan data yang dikandung oleh *nodes*. Murdock (2015) menjelaskan mengenai sebuah panel di dalam aplikasi Maya bernama *Channel Box* yang menampilkan semua informasi *nodes* atau informasi yang dikandung oleh sebuah obyek. Setelah itu, terdapat pula *Attribute Editor* yang menampilkan semua koneksi yang dimiliki oleh obyek tersebut di dalam bentuk peta. Semua atribut yang terdapat di kedua panel tersebut dapat diubah, dikunci, dihapus, dianimasikan, dan ditambahkan oleh pengguna.

2.3.2. *Outliner* dan *Hypergraph*

Maya memiliki beberapa panel yang dapat menampilkan dan mengatur *nodes*, *Outliner* dan *Hypergraph*. Palamar (2016) menjelaskan perbedaan dan fungsi setiap panel-panel tersebut. Koneksi antar *nodes* ditampilkan dengan bentuk visual berbentuk jaringan di dalam *Hypergraph*, sedangkan hierarki *nodes* yang menyusun sebuah tampilan 3D Maya dapat ditampilkan di dalam panel *Outliner*.

2.4. Pivot

Menurut Murdock (2015), lokasi atau titik poros mengatur transformasi sebuah obyek merupakan *pivot point*. Titik ini merupakan manipulator yang mengatur gerak translasi, rotasi, dan ukuran sebuah obyek. Titik *pivot* dapat melakukan transformasi di dalam sebuah axis x, y, atau z di dalam berbagai tampilan Maya. Apabila beberapa obyek diseleksi secara bersamaan, secara otomatis Maya membuat titik pivot di tengah kumpulan obyek tersebut. Hal ini dapat diatur dengan pembuatan *null group*.

2.5. Polygon

Palamar (2016) menjelaskan bahwa obyek tiga dimensi terdiri dari kumpulan *polygon*. *Polygon* terdiri dari titik-titik (*vertices*) yang membentuk garis (*edges*) yang menyusun rangka yang diisi dengan *faces*. *Faces* merupakan penampilan yang tampil di dalam tampilan final atau hasil *render engine*. Susunan dari *faces* membentuk *mesh*. Bentuk *polygon* yang paling umum digunakan merupakan *polygon* empat sisi yang disebut dengan *quad*. Dapat disimpulkan bahwa semua titik membentuk garis, sisi, dan

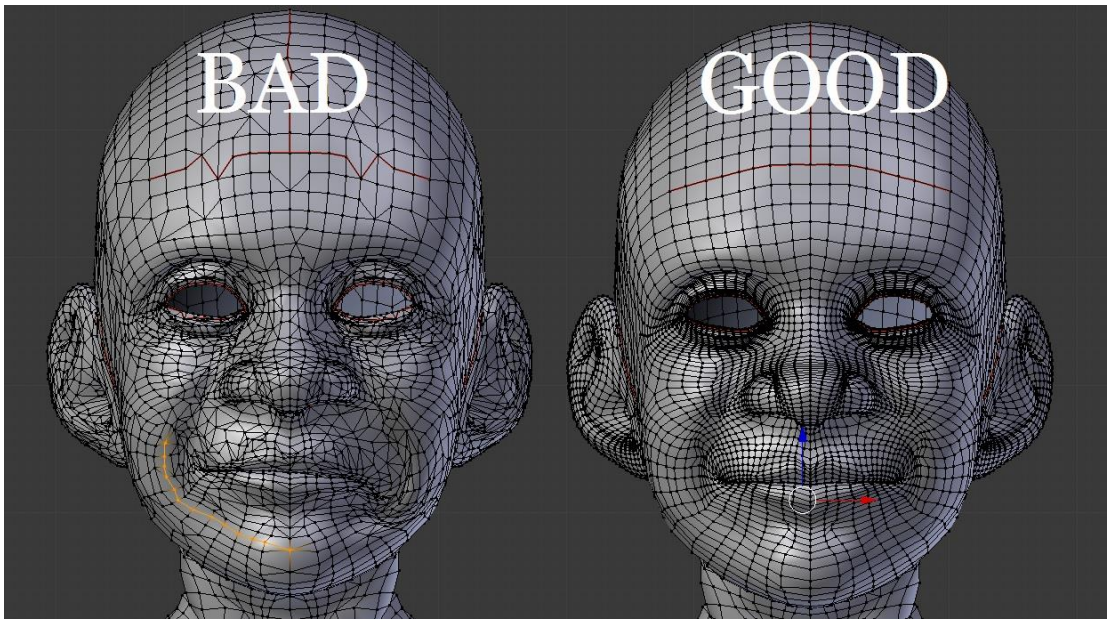
kemudian membuat bentuk tiga dimensi. Setiap komponen dari *polygon* dapat mengalami translasi, rotasi, dan *scale*, kecuali *vertices*.

Kumpulan *vertices* dapat membentuk permukaan yang rata (*planar*) dan permukaan yang bergelombang (*non-planar*). Palamar kembali menambahkan hal-hal yang harus dihindari dalam *polygon*, seperti garis yang bersimpangan dan membentuk sebuah titik (*vertices*) berjumlah lebih dari dua, hal ini akan menyebabkan kesulitan di dalam Program Maya sehingga menghambat proses animasi dan *rigging* (2016).

2.6. Topology

Palamar (2016) menyebutkan *topology* sebagai susunan *polygon edges* di dalam sebuah permukaan. Kemudahan *edges* membuat pengguna dapat merubah susunan *topology* sesuai kehendaknya. Topologi merupakan susunan struktur dari *vertices, edges,* dan *faces*. Susunan topologi yang rapih dan mendetail dapat memiliki banyak keuntungan. Danan (2016) menyebutkan beberapa keuntungan tersebut:

- Susunan topologi yang baik mempermudah modifikasi sebuah model.
- Bentuk dari topologi dapat mempermudah deformasi animasi dikarenakan sususan topologi mempengaruhi lengkungan dan tarikan dari model.
- *Vertices* yang berantakan dapat menumpuk dan membuat tonjolan atau kerutan aneh pada model 3d.
- Tampilan topologi yang terolah mempermudah tampilan *wireframe* dan meringankan data dari komputer dan aliran *workflow* kinerja.



Gambar 2.4. Perbandingan Topologi baik dan buruk

(Danan. (2016). Why Do We Need Topology in 3D Modeling.

http://thilakanathanstudios.com/2016/09/why-do-we-need-topology-in-3d-modeling/)

Selain keuntungan dari susunan topologi yang raph, Danan (2016) juga menyebutkan beberapa hal untuk memastikan susunan topologi yang rapih:

- Selalu perhatikan agar jumlah *vertices* yang digunakan di dalam sebuah model, gunakan sesuai kebutuhan. Jumlah yang terlalu banyak memberikan kerutan bentuk dan deformasi dan jumlah yang terlalu sedikit membuat bentuk yang terlalu lembut atau deformasi yang patah.

- Menyusun *faces* di dalam bentuk *quads* yang terdiri dari empat *vertices*. *Triad* yang terdiri dari tiga *vertices* dapat menyebabkan kerutan pada deformasi bentuk. Selain *triad* terdapat pula bentuk *star* yang merupakan lima titik yang bertemu di tengah. *Star* dapat membuat bentuk penyok atau kerutan.

- Menyusun ukuran *faces* secara merata. Ukuran *faces* yang kecil dan banyak dibutuhkan untuk daerah yang detail dan sebaliknya.

- Memperhatikan susunan *vertices* agar menyusun pola melingkar sesuai dengan susunan *joint* dan daerah gerak pada model.

2.7. Rigging

Palamar (2016) berpendapat bahwa untuk mencapai proses animasi yang mudah dan efisien, diperlukan adanya rancangan *rig*. *Rig* sebenarnya merupakan kumpulan dari *deformers*, *joints*, *expressions*, dan *controls* secara keseluruhan. Di dalam perancangan *rigging*, terdapat hal-hal yang harus diperhatikan menurut Allen & Murdock (2008):

- Jenis-jenis gerakan yang dilakukan oleh tokoh
- Penampilan tokoh
- Bagian tubuh yang digerakkan
- Bagian tubuh yang perlu dispesialisasi.

Setelah memperhatikan hal tersebut, terdapat pula hal penting lainnya yang menentukan sebuah keberhasilan *rig,* O'Hailey (2009) menambahkan terdapat peraturan penting yang harus diikuti oleh setiap rancangan *rig,* yaitu:

- Memperhatikan alur *edgeloops.*
- Tidak pernah menaruh *keyframe* di geometri secara langsung
- Mengunci hal-hal yang tidak disentuh langsung oleh animator.

- Memisahkan geometri, *controller,* dan *joints* di dalam *outliner.*
- Membuat kontroler yang sederhana
- Menjaga atribut awal kontroler di angka nol
- Mengapus semua sejarah transformasi
- Memperhatikan peletakan, arah, dan sudut *joint.*
- Tidak pernah mem-*freeze blendshape*
- *Skinning* yang baik.

2.7.1. Parenting

Di dalam sebuah program, hubungan hierarki orang tua dan anak (*parenting*) dapat diciptakan dengan memilih obyek anak kemudian obyek orang tua, kemudian menekan tombol ‘p’ atau memilh *Edit* dan kemudian *Parent*. Beane (2012) mengingatkan bahwa *parenting* tidak terbatas pada satu anak karena satu orang tua bisa dapat memiliki beberapa anak yang disebut dengan *siblings*. Selain itu, anak dapat menjadi orang tua bagi anak-anak di bawahnya dan seterusnya.

Hierarki *parenting* terlihat seperti pohon keluarga. Hubungan hierarki tersebut terlihat secara jelas di dalam panel *Outliner* pada aplikasi Maya dalam bentuk sebuah grup. Perlu diingatkan bahwa hubungan ini memiliki kelemahan karena perubahan yang dilakukan oleh pengguna kepada orang tua juga mempengaruhi anak, bahkan setelah anak dipisahkan dari grup (Beane, 2012).

2.7.2. Constraints

Selain *parenting*, terdapat sistem lain yang dapat mengontrol obyek secara lebih detail, yaitu *constraints*. Di dalam sistem *constraints*, obyek yang mengikat obyek lain merupakan *driver*, sedangkan obyek yang diikat merupakan *driven*. Allen & Murdock (2008) menyebutkan kembali mengenai beberapa tipe *constraints*:

- *Constraint Point* mengatur hubungan letak di antara dua obyek (*translation*) atau lebih. Oleh karena itu, atribut rotasi tidak dipengaruhi oleh *constraints* tersebut. Palamar (2016) menjelaskan *point constraint* sebagai pengikat data *world space* dari sebuah obyek dengan data *world space* obyek yang lain. Data *world space* merupakan titik koordinat benda di dalam dunia Maya.
- *Constraint Parent* memiliki kemiripan dengan *parenting* biasa. Walaupun begitu, parent *constraint* memiliki beberapa keunggulan seperti mengikat atribut dan bukan mengikat hierarki obyek. Ikatan melalui atribut ini mempengaruhi atribut rotasi dan translasi, namun tidak mempengaruhi atribut tampilan (Allen & Murdock 2008).
- *Aim* mengatur rotasi sebuah obyek untuk selalu menghadap ke obyek yang telah ditentukan. Seringkali *constraints* ini digunakan untuk mata tokoh.
- *Orient* mengatur dan mengikat rotasi obyek satu ke dalam obyek lainnya. Constraint ini memiliki kemiripan dengan *parenting*. Perbedaannya, *driven* dari *orient constraint* memiliki rotasi di dalam porosnya sendiri dan mengikuti jumlah rotasi obyek yang ditentukan. *Parenting* membuat anak (*driven*) berputar mengikuti poros orang tuanya.

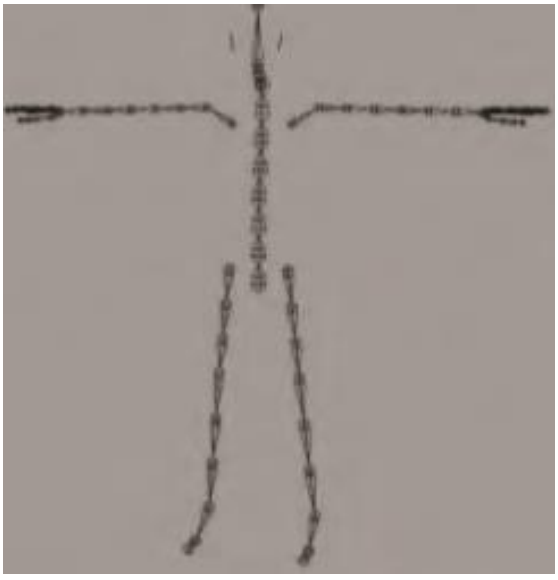
- *Pole Vector* memberikan batasan *vector* agar IK dapat dilekukkan sesuai dengan keingingan pengguna. *Constraints* ini seringkali digunakan untuk lengan dan kaki dimana *pole vector* mengatur ke arah mana lutut atau siku dibengkokkan. Proses pengikatan dapat dicontohkan dengan busur yang diikat dengan tali dan *pole vector* merupakan tali yang mengikat arah lengkungan busur tersebut.
- *Scale* mengatur ukuran sebuah obyek untuk mengikuti ukuran obyek yang telah ditentukan. Data ukuran obyek yang mempengaruhi merupakan presentase perubahan yang dialami. Apabila *controler* bertambah besar tiga kali lipat, begitu pula dengan model sebagai anak.

2.8. Expressions

Expressions dapat mengatur *attribute* dari sebuah obyek sesuai dengan perintah yang diketik oleh pengguna. Beane (2012) memberikan contoh bahwa pengguna dapat mengatur translasi Y sebuah kubus melalui translasi X dari bola. Secara umum, Expression berfungsi sama dengan *constraints*. Walaupun begitu, *constraint* mengikat atribut dengan membuat angka dan perhitungan dari anak yang merupakan hasil salin orang tua dengan sama persis. Walaupun begitu, pengguna dapat mengatur pengaruh dari orang tua terhadap anak dengan lebih detail dan rumit dengan menggunakan *expression*. *Expression* juga dapat menjangkau lebih banyak anak dari *constraint*.

2.8.1. Skeleton

Skeleton merupakan susunan sistem kerangka *joint deformer* yang dapat mengatur bentuk fisik dari komponen geometri terikat (*skinning*) kepada *joints* tersebut. *Joint* berbentuk seperti *sphere wireframe* sederhana yang mewakili sebuah titik virtual. Terdapat jembatan berbentuk *pyramid wireframe* penghubung dua *joints* dengan hierarki *parenting* yang disebut sebagai tulang (*bone*) dengan ujung piramid mengarah ke *joint* anak. (Palamar, 2016).

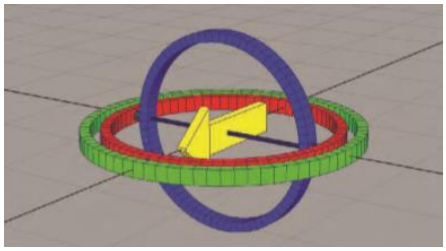


Gambar 2.5. Contoh *Skeleton Joint* Manusia

(Beane, 2012, 3D Animation Essentials, hlm. 182)

Allen dan Murdock (2008) mengingatkan kembali tentang pengaturan orientasi setiap *joints* dan hierarki rotasi mereka. Hierarki rotasi menentukan sumbu dengan pengaruh tertinggi atau dominan (*driver*), dengan sumbu lain yang mengikuti (sebagai *driven*). Pengaturan ini dapat mempengaruhi proses *rigging* untuk menjauhi *gimbal*

lock. *Gimbal Lock* merupakan saat dua sumbu rotasi menyatu menjadi satu dan mempersulit proses animasi.



Gambar 2.6. *Gimbal Lock*

(Beane, 2012, 3D Animation Essentials, hlm. 201)

2.8.2. Kinematics

Terdapat dua jenis gerakan *joint* yang memberikan kesempatan bagi animator untuk melakukan gerakan animasi yang berbeda, berupa *Inverse Kinematics* dan *Forward Kinematics*. *Forward kinematics* menggerakan joint sesuai dengan urutan hierarki yang telah ditentukan di awal. Pengguna dapat menggerakkan bahu, kemudian siku dan lengan, sedangkan inverse kinematics menggerakkan *joint* melalui hierarki paling bawah dan diikuti dengan yang lain (Beane, 2012).

Allen & Murdock (2008) menjelaskan bahwa penerapan *rigging* yang paling efektif didapatkan dari perpaduan dari *Forward* dan *Inverse Kinematics*. Hal tersebut membantu animator dalam menganimasi tokoh dengan pilihan yang lebih luas, kemudahan dari kontrol *IK* atau gerakan yang detail dari *FK*. Perpaduan ini dapat dijangkau melalui program Maya dengan membuat kontrol pengendali *FK* dan *IK* melalui proses *MEL Scripting* atau *Constraint* biasa.

Setelah membahas dasar dari *Inverse Kinematics*, terdapat pula penerapan khusus *IK* untuk bagian tubuh yang memiliki rantai gerakan seperti ekor, rantai, dan tulang belakang. Penerapan ini disebut sebagai *Spline IK* oleh Palamar (2016). *Spline IK* menggunakan curves untuk mengatur rotasi dari setiap joints dan mengatur bentuk rantaian joints tersebut. Allen & Murdock (2008) menjelaskan penerapan ini secara mendetail melalui joint tulang belakang dimana bentuk tulang belakang melengkung mengikuti arah tarikan *controller spline IK*.

2.9. Deformers

Berdasarkan pendapat Palamar (2016), *deformer* dapat digunakan untuk mengatur bentuk (*shape node*) dari sebuah obyek, Setiap obyek yang memiliki control *vertices* dapat menampung jumlah *deformers* yang tidak terbatas karena deformers dapat diaplikasikan ke setiap *faces*, *edges*, dan *vertices*. O'Hailey (2013) kemudian menjelaskan *non-linear deformer* yang seringkali digunakan di dalam software 3D:

2.9.1. Squash and Stretch

Squash dapat membuat obyek menjadi *squash* atau pipih dan menipis saat diberikan tekanan ke tanah atau stretch saat diberikan tarikan dari dua arah yang berlawanan.

2.9.2. Wave atau *Sine*

Wave bekerja sesuai dengan posisi pivot atau controller dari obyek. *Deformer* ini dapat mengatur *amplitude*, *wavelength*, dan *offset*. *Amplitude* merupakan tinggi gelombang dari *handle*. *Wavelength* merupakan panjang satu gelombang, semakin besar

wavelength semakin sedikit gelombang yang dibuat. Palamar (2016) menambahkan pengertian bahwa *wave deformer* membuat rangkaian gelombang seperti efek riak air yang memiliki arah. Selain dari *wave*, terdapat pula sine deformer yang juga berperan untuk merubah bentuk obyek mengikuti gelombang sine.

Walaupun mirip, *wave* dan *sine* memiliki beberapa perbedaan, yaitu di dalam *axis* yang digunakan. Autodesk (2019) menjelaskan bahwa *sine deformer* bergerak searah dengan axis Y dengan *amplitude* yang mengikuti axis Z, sedangkan *wave deformer* bergerak dengan radius dari *axis* X dan Z dan menggunakan *axis* Y sebagai *amplitude*.

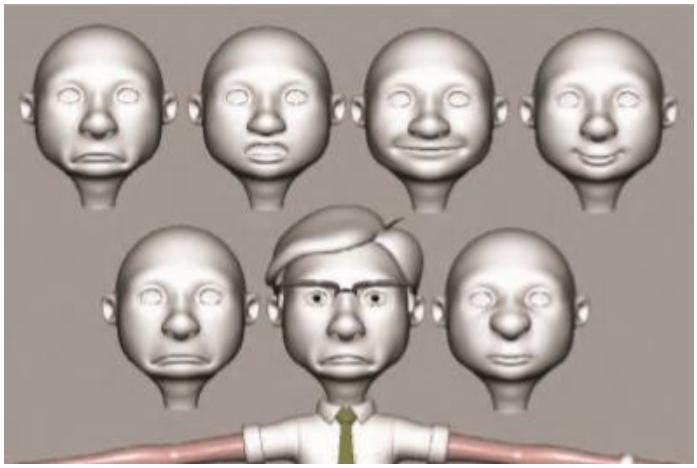
2.9.3. Cluster

Cluster deformer adalah titik-titik deformer yang mengontrol beberapa titik dari obyek seperti CVs, *vertices*, atau titik *lattice*. Titik-titik tersebut diletakkan di sekitar geometri obyek dan mengikat geometri tersebut. Setiap perubahan yang terjadi pada cluster handles mempengaruhi bentuk obyek. Kontrol ini dapat berfungsi sampai ke dalam detail presentasi setiap titik yang dikontrol. Pengguna dapat mengatur berat pengaruh cluster dalam manipulasi rotasi, translasi, dan ukuran setiap *point*. (Autodesk Inc, 2019).

2.9.4. Blendshapes

Berdasarkan Beane (2012), b*lendshapes* merupakan sebuah *deformers* yang menghubungkan dua obyek dengan jumlah *vertices* yang sama dan membuat bentuk berdasarkan perpaduan dari kedua obyek tersebut. *Deformers* ini dapat berkerja untuk

lebih dari dua obyek target. Walaupun seringkali *blendshape* digunakan untuk menganimasikan ekspresi muka dari sebuah tokoh, *blendshape* juga digunakan untuk memperbaiki atau merapihkan geometri yang telah di *skinning*.



Gambar 2.7. Penerapan *Blendshape* Dalam Ekspresi Wajah

(Beane, 2012, 3D Animation Essentials, hlm. 189)

Terdapat beberapa poin penting yang ditambahkan oleh Gend (2017) untuk diperhatikan dalam melakukan proses *blendshape:*

- Geometri final dan geometri yang ditentukan harus memiliki jumlah titik yang sama. Perbedaan dari jumlah titik tersebut dapat mempengaruhi deformasi.
- Geometri yang ditentukan sebagai pilihan untuk *blendshape* tidak boleh melakukan proses *freeze transformation*. Hal ini akan menyebabkab geometri final bergerak ke titik nol di dalam *world space.*
- *Blendshape* bersifat sangat sensitif karena Maya menyimpan setiap data perubahan dan pergerakan dari setiap titik.

2.10. Scripting

Murdock (2015) mendefinisikan *Maya Expression Language* (MEL) sebagai bahasa pemograman yang digunakan untuk mengatur semua kegiatan dan kejadian di dalam Program Maya. MEL dapat langsung diterapkan di dalam program Maya melalui *command line* yang terletak di bawah kiri *interface* Maya atau script editor untuk penggunaan kode yang rumit dan panjang.

2.11. Controls

O'Hailey (2013) menetapkan beberapa aturan di dalam *rigging*. Salah satunya merupakan 'Jaga geometri, *controllers*, dan *skeleton* terpisah satu sama lain'. *Controllers* berfungsi untuk menghemat waktu animator, baik untuk yang mengerti maupun yang tidak mengerti rigging. Sudah menjadi tugas seorang *rigger* untuk meletakkan *icon* di atas geometri agar animator dapat langsung menggerakkan tokoh yang telah di *rigging* tanpa perlu membuka hierarki yang tertanam di dalam *outliner*.

2.11.1. NURBS

NURBS seringkali digunakan untuk menyusun icon atau *controller*. NURBS atau *Non-Uniform Rational B-Spline* tersusun dari titik-titik yang membentuk *curve points*, kumpulan titik yang membentuk sebuah garis (*curves*) (Palamar, 2016). NURBS merupakan satu-satunya tipe geometri di dalam maya yang dapat membentuk *curves* dan *surfaces*. Beberapa tipe geometri lain merupakan *polygon.*

Autodesk (2016) kembali menjelaskan fungsi *NURBS* untuk membuat berbagai tipe bentuk 3d organik karena *NURBS* memiliki kebutuhan *curves* yang minimal untuk membuat sebuah permukaan. Permukaan (*surfaces)* yang didirikan oleh *NURBS* hanya memiliki dua dimensi yaitu panjang dan lebar. Apabila di dalam titik dengan tiga dimensi memiliki *axis X, Y,* dan Z, titik-titik di dalam *NURBS* memiliki dua dimensi yang berbeda. Dimensi tersebut disebut dengan parameter angka. Parameter angka yang menghitung panjang *curve* merupakan parameter U, sedangkan lebar diwakilkan dengan parameter V.

2.11.2. Driven Key

Driven Key bukan pembuatan *keyframe* yang mengikuti *timeline*, melainkan pengaturan animasi yang diatur untuk mengikuti atribut obyek lain yang telah ditentukan. Misalkan, apabila translasi X obyek A berada di sudut 30^o, berarti translasi X obyek A akan berada di sudut 30^o. Palamar (2016) menyebutkan bahwa driven key seringkali digunakan untuk menghubungkan controller dengan *rig* agar menghasilkan *rigging* yang dapat diautomatisasi.

Automatisasi *rigging* mempermudah rancangan kontrol bagian tubuh yang rumit seperti tangan dengan jari-jarinya. Allen & Murdock (2008) berpendapat bahwa *driven-key* dapat mempermudah animator dengan mengendalikan beberapa *joint* dari satu kontrol.

2.12. Skinning dan *Weight Painting*

Proses pengikatan geometri terhadap *joint* untuk membuat semua perubahan pada *joint* mempengaruhi sebuah geometri disebut sebagai *skinning* atau *binding*. Palamar (2016) membagi skinning menjadi dua tipe, yaitu *smooth binding* dan *interactive skin binding*. Pose *skeleton* ketika mengalami proses binding disebut dengan *bind pose*. Pose tersebut merupakan pose dimana tokoh digerakkan semaksimal mungkin untuk melihat batas dari *rig* yang dirancang.

Palamar (2016) menyebutkan bahwa terdapat beberapa metode *skinning*, yaitu:

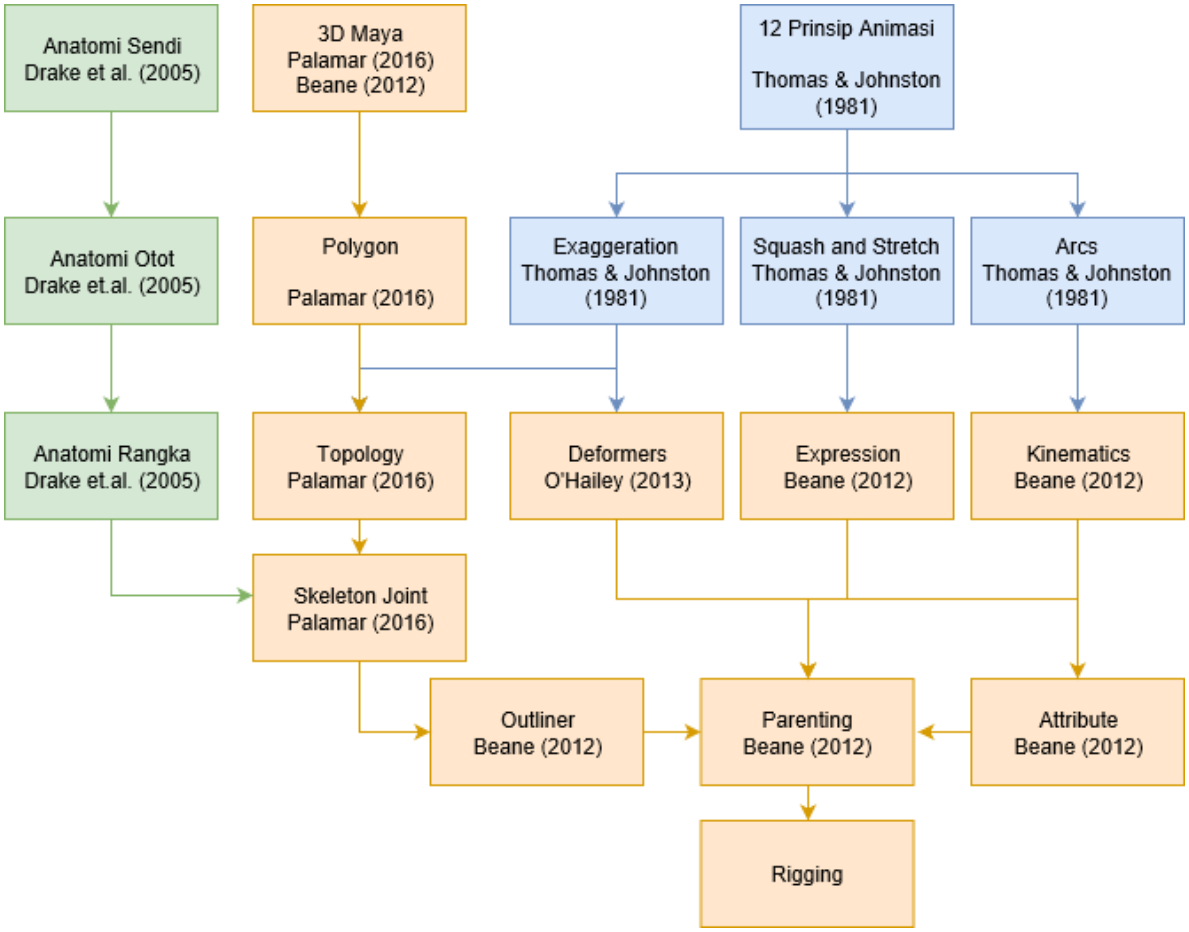
- *Classic Linear* menyebarkan distribusi berat secara merata ke *vertex* geometri di sekitar joint yang ditentukan. Alat ini tidak menyimpan volume geometri yang diikat.
- *Dual Quaternion* memberikan deformasi yang cukup realistik karena menyimpan volume sebuah geometri ketika obyek tersebut mengalami rotasi atau translasi.
- *Geodesic Voxel Binding* merupakan metode pengikatan yang membuat perwakilan *voxel* terhadap *mesh* yang diikat. Metode ini merupakan metode terbaik karena *geodesic voxel* menghitung volume dan bentuk obyek yang diikat. Walaupun begitu, metode ini membutuhkan waktu perhitungan paling lama.

Setelah proses pengikatan antara geometri dengan *joint* dilakukan, terdapat beberapa tahap penting yang harus diperhatikan oleh pengguna. Tidak seperti tubuh manusia yang memiliki otot yang sudah dirancang untuk bergerak, sebuah tokoh tiga dimensi tidak dapat secara otomatis bergerak dengan natural.

Program Autodesk Maya secara otomatis mendistribusikan berat sebuah *joint* ke setiap *vertex* geometri yang terikat dengan *joint* tersebut. Selain *joints*, distribusi berat juga berlaku ke setiap vertex yang dipengaruhi oleh *deformers*. Walaupun begitu, program ini masih memiliki kekurangan tertentu. Pengguna harus merapihkan berat tersebut agar skinning dapat memberikan hasil yang sempurna. Palamar (2016) menyebutkan beberapa peralatan yang dapat membantu proses ini:

- *Interactive Skin Binding Tools* dapat mengatur pengaruh setiap joint terhadap daerah geometri.
- *Paint Skin Weights* mengijinkan pengguna untuk menyempurnakan penyebaran berat secara detail dengan program seperti kuas.
- *Component Editor* mengatur kontrol terhadap besar berat (weight) yang diterima setiap *vertex*.

2.13. Bagan Relasi Antar Teori



Gambar 2.8. Bagan Keterkaitan antar Teori

(Sumber: Dokumentasi Pribadi)