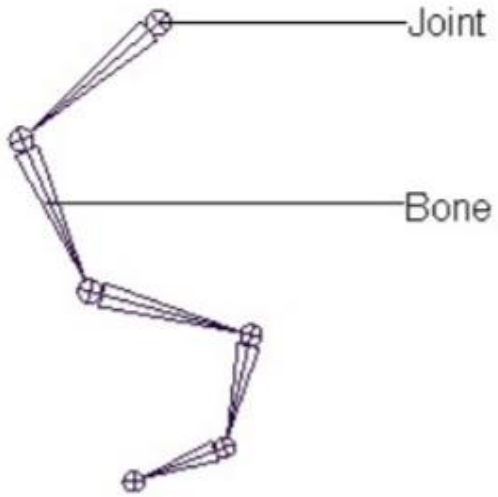
BAB II

TINJAUAN PUSTAKA

2.1. *Rigging*

Tickoo (2016) *Rigging* adalah proses persiapan pada sebuah objek atau tokoh untuk tahap animasi. sebuah *rig* terdiri atas *bones* dan *joints* yang saling berhubungan. Sebuah *joint* adalah sebuah titik yang kemudian menggerakkan *mesh* (geometri dalam 3D) tokoh. Lebih lanjut, sebuah *bone* akan menghubungkan 2 *joints* layaknya sebuah tulang menghubungkan 2 sendi, memungkinkan *joints* pada hirarki yang lebih rendah untuk diputar sesuai poros *joint* penghubung. Sekumpulan dari sistem ini akan membentuk sebuah kerangka/skeleton utuh. Analogi sederhana *rig* bagi *mesh* adalah tulang pada manusia bagi tubuh kita.



Gambar 2.1. Contoh kerangka

(sumber : Autodesk Maya 2017 A Comprehensive Guide)

Dalam sebuah animasi 3D, ada tiga sumbu kedalaman : x, y, dan z. Sumbu "dunia" atau *world axis*, merupakan sebuah sumbu mutlak yang digunakan untuk menghitung lokasi serta rotasi sebuah objek terhadap dunianya. Objek-objek ini, termasuk *joints*, memiliki sumbu mereka tersendiri, dikenal dengan nama *local*

axis. Dengan sumbu ini, setiap sendi dalam sebuah *rig* akan memiliki sumbunya tersendiri, yang memungkinkan pergerakkan individual.

Sebuah sistem persendian diatur dalam sebuah hirarki. Misalnya, *joint* A berada di bawah *joint* B secara hierarkis, maka saat B: berputar, A akan ikut berputar sesuai sumbu B. Dalam kasus ini, B adalah *parent* dari A, dan A adalah *child* dari B. Kecuali dalam kasus khusus, pada umumnya sebuah *joint* akan selalu berada di bawah *joint* lain, sehingga membentuk sebuah kerangka utuh yang saling memengaruhi.

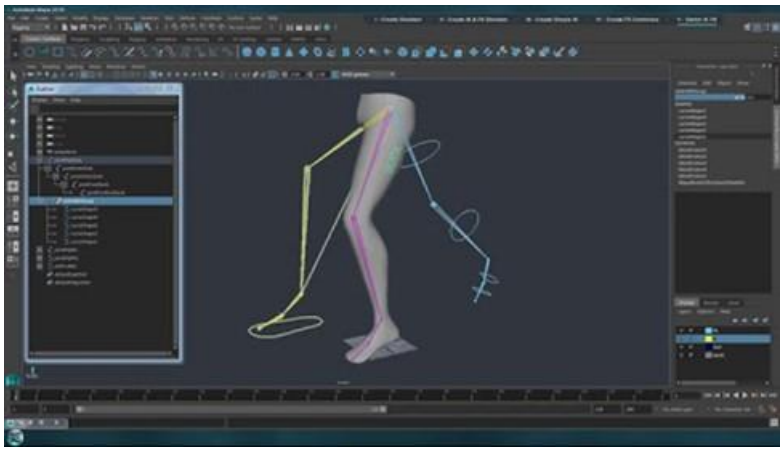
O'Hailey (2013) menjelaskan bahwa ada beberapa peraturan yang wajib diikuti sebelum masuk ke tahap *rigging*. Beberapa peraturan tersebut yang relevan untuk penulisan ini adalah sebagai berikut :

1. Jangan pernah menggerakkan geometri/*mesh* secara langsung.
2. Kunci atribut yang tidak akan dianimasikan.
3. Simpan geometri, *controls*, dan *joints* dalam *group* terpisah.
4. Ciptakan *rig*/*controller* yang masuk akal bagi *animator*.
5. *Atributte control*s dan *joints* harus memiliki posisi netral 0.
6. Selalu hapus *history* pada *rig* (bila memungkinkan) agar *rig* tetap ringan.
7. Tentukan peletakan, serta orientasi *joints*.

2.1.1. *Kinematics*

Cara menggerakkan tulang dengan cara demikian adalah apa yang disebut dengan *rig forward kinematic* (FK). *Kinematics* adalah logika bergerak dalam sebuah rig. Alternatif lain dari FK adalah *inverse kinematic* (IK), dimana *joint* terpilih teratas terhubung dengan *joint* terpilih terbawah. Dengan IK, saat salah satu *joint* digerakkan, maka segala sesuatu di antaranya akan ikut bergerak secara otomatis,

mencari posisi paling efisien dan logis. Begitu juga sebaliknya bila parent yang digerakkan. FK pada umumnya digunakan saat kendali presisi dibutuhkan (misal : kendali leher pada manusia), sedangkan IK lebih umum digunakan saat presisi tidak menjadi prioritas utama atau kalkulasi komputer sudah cukup akurat (misal : kendali lutut).



Gambar 2.2. IK (kiri) dan FK (kanan)

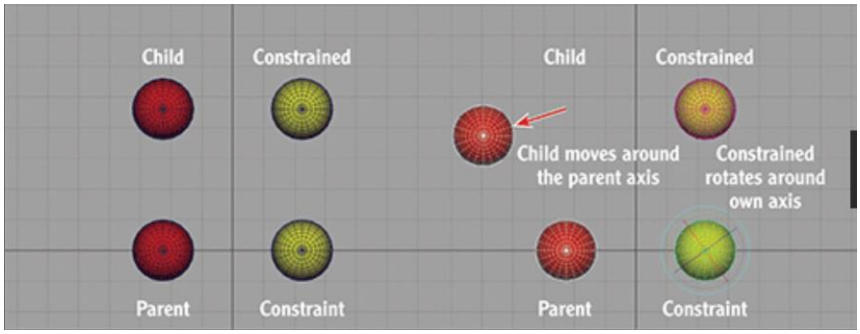
(sumber : www.youtube.com)

2.1.2. *Constraints*

Beanne (2012) Menjelaskan *constraints* sebagai sebuah sistem yang memungkinkan satu objek untuk mengendalikan objek lain. Hal ini memungkinkan sekelompok objek untuk berperan sebgai pengendali objek yang sesungguhnya, menjaga objek utama agar tetap bersih dari *keys*/animasi. Pada umumnya, perangkat lunak 3D memiliki beberapa jenis constraint:

1. *Point constraint* : Mengikat atribut translate atau posisi.
2. *Orient constraint* : Mengikat atribut rotasi.
3. *Scale constraint* : Mengikat atribut ukuran.
4. *Parent constraint* : Mengikat ketiga atribut di atas

5. *Aim constraint* : Membuat satu objek untuk secara otomatis berputar agar selalu mengarah ke objek lain.

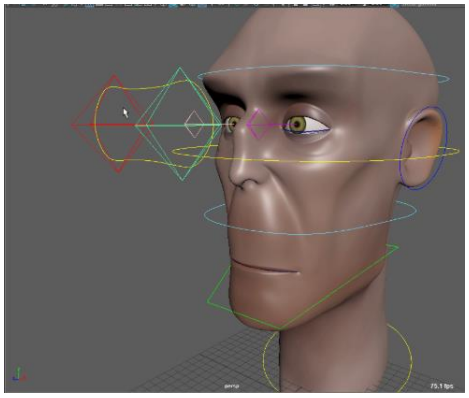


Gambar 2.3. Contoh *constraint*

(sumber : *3D Animation Essential)*

2.1.3. *Controllers*

Sebuah *controller* adalah suatu objek yang berperan sebagai penggerak objek lain atau bagian dari *rig*. Apapun dapat menjadi *controller*, dari poligon, *nurb*, hingga *locator*. O'Hailey (2013) dalam penggunaannya pada *rigging* menjelaskan bahwa salah satu peraturan dasar *rigging* adalah untuk jangan pernah menggerakan geometri/*mesh* secara langsung. Maka dari itu *controller* di sini berperan sebagai peredam agar *rig* dan mesh tetap bersih dari *key*, dengan men-*constraint joints pada controllers*. *Controller* dapat berbentuk apapun, namun sering kali rigger membuat visual yang lebih intuitif guna mempermudah *animator* dalam memahami dan menggunakan *rig*.

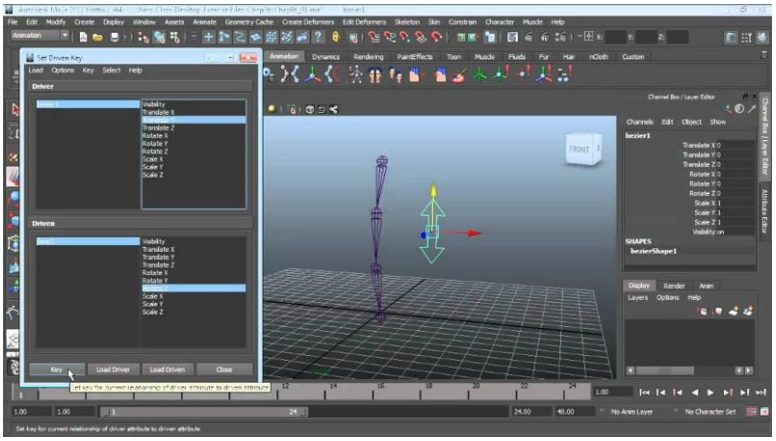


Gambar 2.4. Controller wajah

(sumber : www.bindpose.com)

2.1.4. *Set Driven Keys*

Set driven keys adalah fitur yang menghubungkan suatu atribut pada atribut lain dengan prinsip persamaan. O'Hailey (2013) *set driven keys* adalah seperti *scripting* melalui visual. Dengan merubah angka pada sebuah atribut, maka akan memengaruhi atribut lain sesuai dengan pengaturan yang sudah ditentukan. Jika *driver* (sebutan untuk atribut yang memengaruhi) digerakkan, maka atribut *driven* (yang dipengaruhi) akan berubah. Angka pada atribut-atribut ini tidaklah harus identik.



Gambar 2.5. Jendela driven keys memperlihatkan koneksi posisi Y controller panah dengan rotasi Z joint

(sumber : i.ytimg.com)

2.1.5. *Connection Editor*

O'Hailey (2013) *Connection editor* adalah sebuah fitur yang memungkinkan *rigger* untuk menghubungkan 2 atribut secara langsung. Yang membuatnya berbeda dari *driven keys* adalah satu atribut tidak menggerakkan atribut lain, melainkan atribut tersebut disalin. Jika skala dari objek A yang dihubungkan dengan skala objek B adalah 1, maka objek B akan memiliki skala 1 juga.

2.1.6. *Deformers*

Tickoo (2016) *deformer* adalah sebuah fitur guna merubah bentuk *mesh* objek. Hal ini dilakukan tanpa benar-benar merubah objek, melainkan seperti sebuah efek pada objek yang memengaruhi verteks (titik, unit terkecil pada sebuah *mesh*). Ada beberapa jenis *deformer*, namun hanya ada tiga yang relevan bagi skripsi ini :

1. *Linear deformer (squash)* : memipihkan dan memanjangkan objek
2. *Blendshapes* : menyalin posisi verteks dari pose-pose yang sebelumnya telah ditentukan.
3. *Cluster deformer* : pengelompokkan objek/komponen objek tertentu yang kemudian diletakkan di bawah sebuah titik.

2.1.7. *Skinning*

O'Hailey (2013), adalah proses pelekatan *mesh* pada *skeleton*, agar bergerak mengikuti *skeleton.* Proses inilah yang memungkinkan *rig* yang telah dibuat agar dapa mengendalikan *mesh* tokoh/objek. Proses ini menentukan pengaruh tiap *joint* terhadap tiap vertex pada *mesh.* Pada awal tahap *skinning*, perangkat lunak animasi pada umumnya akan berusaha sebaik mungkin untuk menebak dan mengikat verteks pada *joint* terideal. Namun, proses ini tidaklah sempurna, sehingga ada

baiknya untuk memeriksa kembali dan bila diperlukan, mengatur secara manual pengaruh *joint* yang bersangkutan. Proses pengaturan dan penyempurnaan *skinning* ini disebut dengan *weight painting*.

2.2. *nParticles*

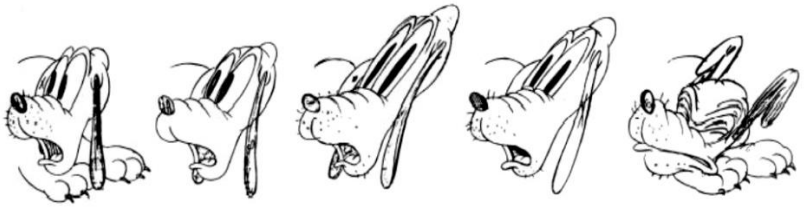
Beanne (2003) menjabarkan partikel sebagai titik di ruang tiga dimensional. Huruf "n" pada *nParticles* merupakan sebuah singkatan untuk "*nucleus*", nama dari sistem simulasi yang digunakan. Pada dasarnya, informasi titik tersebutlah yang digunakan *animator* dengan cara memberinya *mesh*, efek, atau lainnya. Di luar sistem *nucleus*, terdapat sistem *field/solver* yang pada dasarnya berperan sebagai efektor yang dapat berinteraksi dengan sistem *nucleus. Field/solver* dapat digunakan untuk membuat angin tambahan, titik gravitasi, menahan pose, dan lain-lain.

Seperti yang sudah disebutkan sebelumnya, sistem partikel ini dikendalikan oleh sebuah sistem simulasi, yang/sehingga membuatnya reaktif terhadap hal lain seperti objek dalam satu sistem yang sama, angin, gravitasi, dan lainnya. Informasi titik-titik ini hanyalah data yang tidak memiliki bentuk fisik yang dapat dilihat. Beranjak dari sana, *animator* kemudian menggunakan data ini untuk diberikan efek seperti air, awan, debu, dan lainnya, merubahnya menjadi *mesh*, atau mengikatkan *mesh* yang sudah didesain sebelumnya. Dengan ini, partikel memiliki bentuk fisik yang dapat kemudian menjadi hasil *output* akhir.

2.3. *Squash and Stretch*

Thomas dan Johnston (1981) menjelaskan bahwa dalam animasi terdapat 12 prinsip yang muncul akibat para *animator* yang terus mencari metode baru dalam menggambar dengan hasil yang konsisten. Salah satu dari ke-12 prinsip tersebut adalah *squash and stretch*, atau memipih dan memanjang. Prinsip ini melebih-

lebihkan deformasi sebuah objek guna membuat pergerakkan memberi sebuah kesan pada penonton. Selain itu, prinsip ini juga digunakan dalam memberi antisipasi serta gerakan sekunder pada sebuah objek atau tokoh.



Gambar 2.6. Controh *Squash and stretch*

(sumber : Walt Disney)