# BAB II

# TINJAUAN PUSTAKA

## 2.1 Animasi

Pranowo (2011) dan Wright (2005) menjelaskan bahwa animasi berasal dari bahasa latin anima, yang jika diterjemahkan ke dalam bahasa Indonesia berarti jiwa, dan animare yang berarti nafas kehidupan. Dalam bahasa Inggris, animasi berarti memberikan kehidupan atau menggerakkan. Saat kita melihat animasi, kita melihat karakter atau objek yang terlihat seolah – olah hidup, namun pada kenyataannya tidak, dengan demikian animasi merupakan ilusi yang diciptakan menggunakan gerakan – gerakan gambar sehingga karakter tersebut terlihat hidup. Thomas & Johnston (1981) menambahkan dua prinsip animasi yang kerap digunakan ialah *squash and stretch* dan *exegration*.

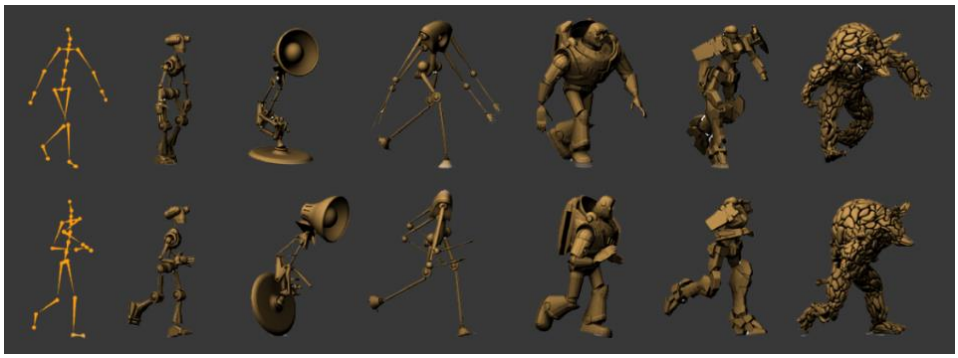
## 2.2 Animasi 3D

Aditya (2009) menjelaskan bahwa animasi 3D merupakan animasi yang memiliki wujud tiga dimensi. Hal ini diperjelas oleh Beane (2012) yang menjelaskan bahwa animasi 3D adalah animasi yang diproduksi dengan menggerakkan objek 3D yang dibentuk di dalam *software* komputer seperti Maya, 3DS Max, Blender, Houdini, C4D, Marvelous Designer, dan *software* – *software* 3D lainnya.

## 2.3 Rigging

Satriawan (2016) menjelaskan bahwa *rig* pada dasarnya adalah kerangka digital yang mengikat *mesh*. Pernyataan ini didukung oleh Aditya (2009) dan Beane (2012)

yang menjelaskan *rigging* adalah proses pemasangan tulang, *rigging setup*, beserta *controllerny* ke dalam tokoh animasi 3D *digital* supaya tokoh tersebut bisa digerakkan menggunakan sistem *rig* tersebut.



Gambar 2.1. *Rig* dengan pose tokohnya

(https://kaylah3dmodelling.wordpress.com/2015/03/04/rigging/)

Rigging merupakan proses yang sangat penting, karena kualitas animasi tokoh sangat ditentukan oleh bagus tidaknya *rig* yang mengontrolnya. Hal ini dipertegas oleh Allen & Murdock (2008) yang menyatakan bahwa membuat *rig* tokoh sama halnya dengan membangun fondasi rumah yang kokoh, dengan fondasi yang kokoh *rig* bisa ditambah berbagai macam fitur seperti otomatisasi gerakan dan lain – lainnya, namun, jika fondasinya tidak kokoh, maka *rig* akan sangat sulit digunakan. *Rig* pada umumnya berisikan *joint*, *handle*, dan *controller*, namun setiap tokoh memiliki prosedur *rigging* yang berbeda – beda, prosedur *rigging* yang digunakan juga akan sangat dipengaruhi oleh gerakan sperti apa yang ingin dicapai oleh tokoh yang akan di*rig*.

Allen & Murdock (2008) menerangkan bahwa sebelum memasuki proses *rigging*, sebaiknya *rigger* merancang sistem *rig* terlebih dahulu. Proses membuat

perancangan sistem *rig* bisa memakan waktu yang cukup lama. Namun, perancangan tersebut akan sangat menghemat waktu proses *rigging*, dikarenakan *rigger* sudah paham prosedur seperti apa yang dibutuhkan untuk me*rig* tokohnya. Beberapa hal yang perlu diperhatikan *rigger* saat merancang sistem *rig*, yaitu:

1. Daftar gerakan – gerakan yang akan dilakukan oleh tokoh tersebut
2. Bentuk tokoh
3. Daftar bagian tubuh yang akan bergerak
4. Daftar bagian tubuh yang perlu *rig* khusus

Allen & Murdock (2008) menjelaskan bahwa salah satu aspek yang sangat penting dalam proses *rigging* adalah memahami *rig* seperti apa yang dibutuhkan *animator*. Jika *rig* yang dibuat dapat dipahami *animator* dengan mudah mereka akan bisa mengerjakan animasi dengan baik dan cepat. Namun, jika *rig* yang dibuat sangat berantakan sehingga *animator* tidak memahami kontrol apa yang menggerakkan bagian tubuh apa, maka pekerjaan mereka akan menjadi sangat terhambat, dan yang akan terjadi adalah *rig* akan dikembalikan kepada *rigger*, untuk diperbaiki atau *rigger* akan dimintai keterangan mengenai *rig*nya. Hal ini akan sangat menghambat jalannya produksi dan mengacaukan *timeline* produksi yang harusnya dipatuhi semua divisi. ada tiga hal yang perlu diperhatikan saat membuat sistem *rig* untuk tokoh, yaitu:

1. **Keorganisiran, kerapihan, dan kemudahan digunakannya *Rig.***

   Hal ini meliputi penamaan di sistem *rig* yang meliputi penamaan *mesh, joint, Ik, curve, controller;* dan hirarki sistem *rig*.

2. **Kefektifan penempatan *Controller.***

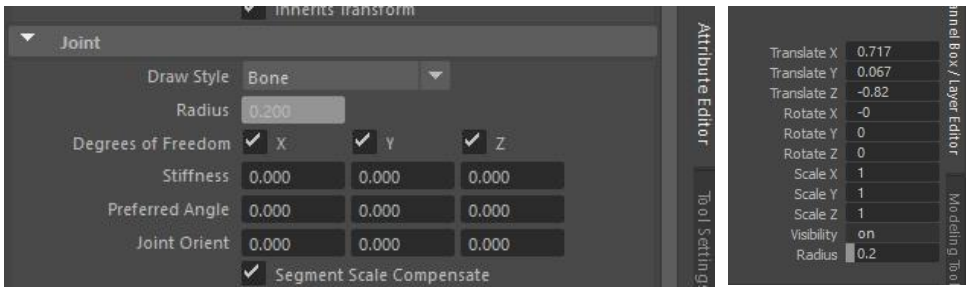
   Hal ini meliputi penempatan pivot *controller* yang sesuai dengan anggota tubuh yang digerakkan.

3. ***Interfaces* yang komunikatif.**

   Hal ini meliputi bentuk *controller* dan *Graphic User Interface* yang mudah dipahami.

## 2.4 Rigging Setup

*Rigging setup* tidak lain adalah proses penempatan *joint – joint* yang dibutuhkan ke dalam *mesh* tokoh, seperti yang dijelaskan Aditya (2009). Beane (2012) menambahkan bahwa dalam menyusun *joint* hendaknya memperhatikan *pivot joint* tersebut. *Pivot* sendiri adalah titik dimana objek tersebut berotasi. Allen & Murdock (2008) menambahkan bahwa dalam penempatan *joint*, sangat disarankan untuk membiarkan rotasi joint tetap 0. Hal ini diperuntukkan menjaga *joint* lebih rapih, dan bersih. Namun, jika joint harus dimiringkan, maka yang harus diubah pengaturannya adalah *joint orient*nya. Meskipun *joint orient* diubah, *joint* akan tetap terlihat rapih dan bersih, karena *joint orient* tidak terlihat di *channel box*.




Gambar 2.2. *Atribute editor joint*

(dokumentasi pribadi)

Proses ini pada umumnya dimulai dari badan, lalu tangan atau kaki, kemudian anggota tubuh tambahan seperti sayap, ekor dan lain lainnya.

### 2.4.1. *Setup Rig* pada Badan

Allen & Murdock (2008) menjelaskan bahwa *rig* pada bagian badan merupakan fondasi utama *rig* tokoh, yang mengalir dari tulang panggul hingga tulang leher. *Rig* pada bagian badan akan menggerakkan keseluruhan tokoh. Jika badan tokoh di*rig* dengan baik, maka tokoh tersebut bisa dianimasikan dengan lebih leluasa.

Allen & Murdock (2008) menambahkan setiap tokoh memiliki kebutuhan peranjcangan *rig* badan yang berbeda. Contohnya, tokoh *Spiderman* membutuhkan *root joint* yang berada di bagian lehernya, saat animasi spiderman mengayun – ayun di udara. Hal ini untuk memudahkan animasi gerakan ayunan badan *Spiderman*. Saat Spiderman menyentuh tanah, *root joint* diperlukan kembali ke bagian tulang panggul, agar lebih mudah untuk menganimasikan *walk cycle*. Perancangan *rig* sendiri ada beberapa jenis, dua diantaranya adalah *Classic Body Root* dan *Free Root*.

1. Pada *Classic Body Root, root joint* diletakkan di bagian panggul. *Classic body root* cenderung lebih mudah dibuat, dan cocok untuk gerakan – gerakan yang simpel, dan sangat stabil, namun *animator* akan kesulitan menganimasikan panggulnya.
2. Pada *free root*, *root joint* diletakkan diatas *root panggul*. *Free root* cenderung lebih sulit dibuat, namun *animator* bisa dengan leluasa menganimasikan gerakan panggul, karena *joint* panggul bukan

merupakan *root joint*nya, sehingga cocok untuk animasi yang lebih rumit seperti berjalan dan berlari.

### 2.4.2. *Setup Rig* pada Bahu dan Lengan

Allen & Murdock (2008) menjelaskan bahwa *rig* pada bagian lengan dan bahu merupakan bagian *rig* yang rumit, terlebih di bagian tulang selangka yang banyak gerakannya. *Setup Rig* pada bagian bahu sendiri dibagi menjadi dua, yaitu *simple shoulder setup* dan *clavicle – shoulder setup*. Keduanya memiliki kelebihan dan kekurangannya masing masing.

1. Pada *simple shoulder setup, setup* cenderung lebnih simpel, dan cepat, sehingga akan sangat efektif untuk tokoh sampingan.
2. Pada *clavicle – shoulder setup, setup* cenderung lebih kompleks, namun pergerakan yang bisa didapat akan lebih realistis, dan memungkinkan tokoh untuk mengangkat bahu.

### 2.4.3. *Setup Rig* pada Kaki

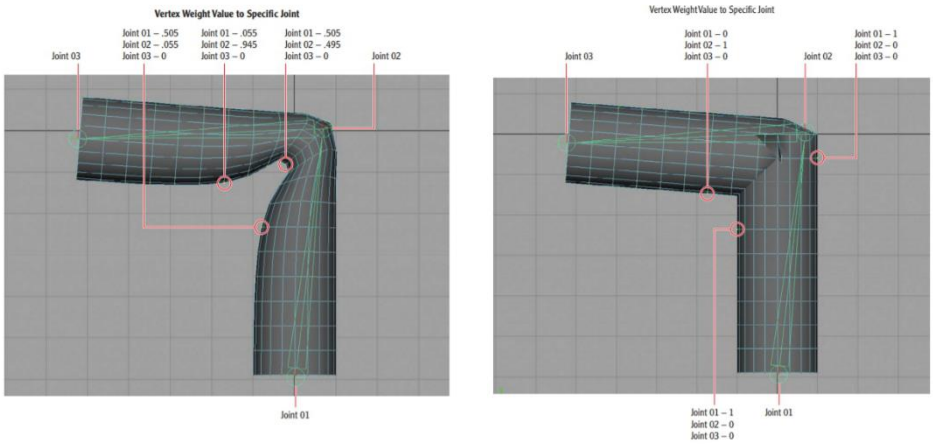
Allen dan Murdock (2008) menjelaskan bahwa, dalam perancangan *joint* pada kaki ada dua aspek dasar yang perlu diperhatikan, yaitu *classic* IK *leg* dan *advanced leg*. Keduanya memiliki kelebihan dan kekurangannya masing masing.

1. Pada *classic* IK *leg*, *setup* dan penggunaannya cenderung lebih mudah. Namun, *setup* hanya sebatas IK saja.

2. Pada *advanced leg*, *setup* cenderung lebih rumit karena menggabungkan IK dan FK, namun, itu akan memberikan lebih banyak pilihan untuk *animator*.

## 2.5 Skinning

Allen & Murdock (2008) menjelaskan bahwa *skinning* adalah proses mengikatkan *joint – joint* dengan *mesh* yang bersangkutan, sehingga saat *joint* digerakkan, bagian tubuh *mesh* yang bersangkutan juga ikut bergerak. Beane (2012) menambahkan, pada dasarnya, *joint* akan mengikat *mesh* melalui *vertex – vertex*nya. Sama halnya dengan *constraint*, satu *vertex* bisa terikat beberapa *joint* sekaligus, hal tersebut dinamai *falloff*.




Gambar 2.3. Efek *skin weight* terhadap *mesh*

(Beane, 2012, hlm 187 & 188)

Setelah *mesh* diikat dengan *joint*, perlu dilakukan proses *paint skin weight*, proses ini dilakukan untuk mengatur sebanyak apa *skin weight* tiap *vertex*, dengan menggunakan metode *digital painting*. Setiap vertex memiliki 100% skin weight

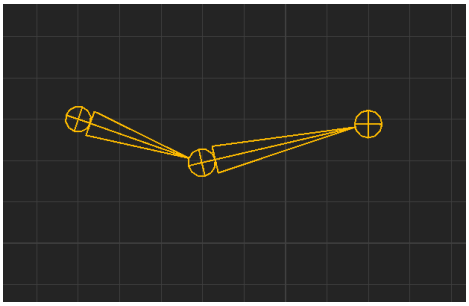
yang direpresentasikan dengan angka 1, tidak akan kurang atau lebih. *Skin weight* tersebut dapat dibagi – bagi ke joint – joint yang mengikatnya.

## 2.6 Sistem Gerak dalam *Rigging*

Allen & Murdock (2008) dan Beane (2012) menjelaskan bahwa, sistem gerak dalam *rigging* pada umumnya dibagi menjadi dua yaitu FK (*Forward Kinematics* ) dan IK (*Inverse Kinematic*). Keduanya memiliki kelebihan dan kekurangannya masing – masing.

### 2.6.1. FK (Forward Kinematics)

Setelah tulang – tulang atau *joint* dibentuk, *joint – joint* tersebut pada dasarnya digerakkan dengan sistem gerak FK atau *forward kinematics*. Dalam sistem gerak ini, hirarki yang di atas atau induk *joint* akan menggerakkan hirarki yang di bawahnya atau anak *joint*. Namun, saat anak *joint* digerakkan induk jointnya tidak akan ikut bergerak. Dengan kata lain, saat *joint* bahu digerakkan, *joint* siku dan pergelangan tangan akan ikut bergerak juga. Namun, saat *joint* pergelangan tangan digerakkan, *joint* siku dan bahu tidak akan ikut bergerak. Metode ini cocok untuk menganimasikan gerakan seperti melambaikan tangan, mengayunkan kaki, dan gerakan - gerakan lainnya yang hanya perlu memutar *joint* secara FK.



Gambar 2.4. *Forward kinematics*

(dokumentasi pribadi)

Allen & Murdock (2008) meanmbahkan bahwa sistem gerak FK sangat tidak cocok untuk memposisikan pergelangan ke tempat tertentu, terlebih menganimasikannya. Salah satu contohnya yaitu menganimasikan gerakan membuka gagang pintu. Posisi tangan dan jari – jarinya yang sudah sesuai akan berubah saat *joint* bahu dan siku digerakkan, sehingga pergelangan tangan harus diposisikan lagi di setiap frame. Hal ini, sangatlah tidak efektif, dan membuang banyak waktu untuk gerakan yang mudah, dan simpel.

### 2.6.2. IK (Inverse Kinematics)

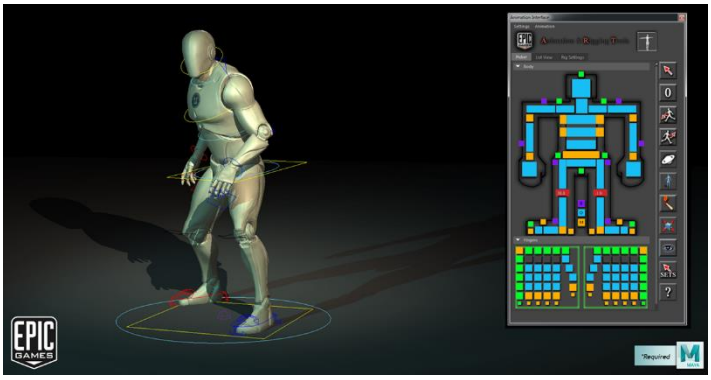
IK adalah sistem gerak bekerja berkebalikan dengan FK yang memungkinkan *joint* hirarki di bawah atau anak *joint* menggerakkan *joint* hirarki di atas atau induk *joint*. Sistem gerak IK biasanya dipakai pada anggota tubuh seperti tangan, kaki, bahkan beberapa *rig* menggunakan IK sebagai sistem gerak badan, ekor, dan leher.

Dengan sistem gerak IK, akan sangat mudah memposisikan anggota – anggota tubuh tersebut di posisi tertentu. Setelah diaplikasikan, IK akan menanamkan efektor di anak *joint* yang dipilih. Dengan menggerakkan efektor

tersebut, anak *joint* akan mengikutinya, dan induk *joint* akan ikut bergerak. Saat diaplikasikan, IK juga akan menambahkan garis yang menghubungkan induk dan anak *joint*nya.

## 2.7 Controller

Allen & Murdock (2008) menjelaskan bahwa *controller* adalah salah satu bagian terpenting dalam *rig* yang merupakan bentuk visual, baik itu berupa kurva, *locator*, *plane*, atau *polygon* yang berfungsi menggerakkan bagian – bagian tubuh yang di*rig*. Dengan kata lain, *controller* menggerakkan *joint – joint* yang sudah ditanam dalam *mesh*. Setelah *Controller* dipasang, sangat diharuskan untuk membuat posisi dan rotasi *controller* menjadi 0, 0, 0. Ini untuk memudahkan *animator* mengembalikan posisi *controller* menjadi seperti semula.



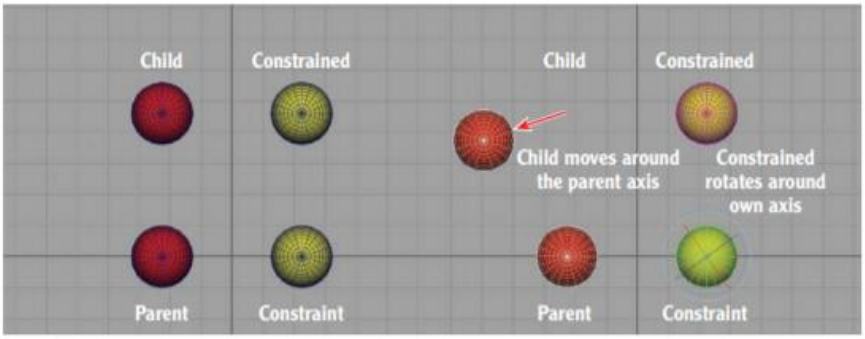

Gambar 2.5. Satu set *controller*

(https://www.unrealengine.com/en-US/blog)

## 2.8 Constraint

Beane (2012) menjelaskan bahwa *constraint* adalah fitur yang berfungsi untuk menghubungkan pergerakan sebuah objek dengan objek lainnya. Allen & Murdock

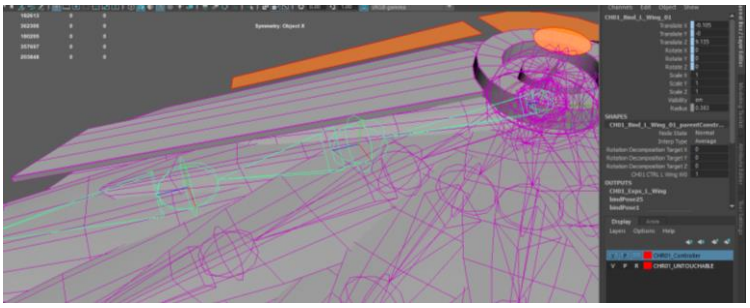
(2008) menambahkan bahwa *constraint* memungkinkan untuk menggerakkan suatu objek dengan menggerakkan objek lainnya, mirip dengan sistem hirarki induk dan anak. Namun, dalam *constraint* tidak terlihat hirarki sama sekali di *outliner*.




Gambar 2.6. *Constraint*

(Beane, 2012, hlm 191)

Setelah diaplikasikan, objek yang di*driven* akan ditambahkan *node constraint* yang berisikan informasi mengenai *constraint* yang dihubungakn dengan objek *driver*nya. Informasi tersebut juga bisa dilihat di *channel box* objek *driven*. Setelah diaplikasikan kotak *translate, rotate, scale,* atau *pole vector* dalam *channel box* ataupun *atribute editor* akan berubah menjadi biru dan akan dikunci.



Gambar 2.7. Efek *constraint* pada atribut

(dokumentasi pribadi)

Dalam proses *rigging*, *constraint* biasanya digunakan untuk menggerakkan *joint, mesh,* atau objek lainnya secara otomatis dengan menggerakkan *controller*nya.

Untuk mengaplikasikan *constraint*, objek *driver* harus diselect terlebih dahulu, lalu objek *driven*, setelah itu *constraint* bisa diaplikasikan. satu objek *driver* bisa menggerakkan beberapa objek *driven* sekaligus. Begitu pula sebaliknya, satu objek *driven* bisa digerakkan beberapa objek driver sekaligus. Constraint sendiri terdiri dari beberapa jenis, sesuai dengan fungsinya masing – masing, yaitu:

1. ***Point Constraint***

   *Point constraint* berfungsi untuk menghubungkan hanya *atribute translate* objek *driven* dengan *driver*, sehingga, jika objek *driver* dipindahkan, maka objek *driven* akan ikut berpindah.

2. ***Orient Constraint***

   Orient Constraints berfungsi untuk menghubungkan hanya *atribute rotate* objek *driven* dengan *driver*, sehingga, jika objek *driver* diputar, maka objek *driven* akan ikut berputar.

3. ***Scale Constraint***

   *Scale Constraints* berfungsi untuk menghubungkan hanya *atribute scale* objek *driven* dengan *driver*, sehingga, jika objek *driver* diperbesar atau diperkecil, maka skala objek *driven* akan ikut berubah.

4. ***Parent Constraint***

   *Parent Constraints* pada dasarnya gabungan dari *point constraint* dan *orient constraint*, yang berfungsi untuk menghubungkan *atribute translate* dan

*rotate* objek *driven* dengan driver, sehingga jika objek *driver* dipindah dan diputar, maka objek *driven* akan ikut berpindah dan berputar.

5. ***Aim Constraints***

   *Aim Constraints* berfungsi untuk mebuat objek *driven* selalu menghadap ke arah objek *driver*, sehingga, jika objek driver dipindahkan, maka objek *driven* akan berputar menghadap objek *driver*. Biasanya dipakai untuk mengarahkan arah bola mata.

Gambar 2.8. Pengaplikasian *aim constraint*

(Beane, 2012, hlm 190)

6. ***Pole Vector***

   *Pole Vector* pada dasarnya dipakai pada IK, yang berfungsi untuk menghadapkan *vector* IK *driven* ke arah objek *driver*. Biasanya dipakai untuk mengarahkan siku dan lutut.

## 2.9 Attribute Editor & Channel Box

Allen & Murdock (2008) menjelaskan bahwa *Attribute Editor* merupakan fitur yang berfungsi untuk mengakses atribut – atribut yang tersedia pada objek yang diseleksi. Pada objek dasar, atribut – atribut yang tersedia meliputi *transform node, shape node, history node,* dan lain – lainnya.

Sama halnya dengan *Attribute Editor*, Hanya saja, tidak sebanyak *Attribute Editor*. Dalam Proses *rigging*, *rigger* bisa menambahkan atau mengurangi jumlah atribut yang bisa diakses melalui *Channel Box*. Tujuannya tidak lain adalah untuk memberi tahu *animator* atribut mana saja yang boleh diakses, dan mana yang tidak boleh.

## 2.10 *Node* Atribut

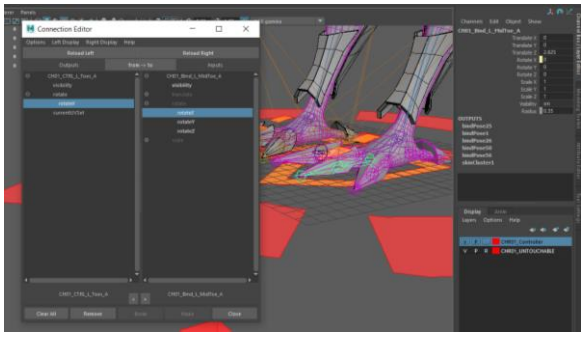
Allen & Murdock (2008) menjelaskan bahwa, *rigger* pasti membutuhkan atribut baru untuk mengotomatisasi gerakan tokoh, seperti mengangkat tumit, mengepalkan tangan, dan gerakan lainnya. Atribut tersebut dapat dibuat menggunakan fitur *Add Attribute*, dengan fitur tersebut, *rigger* rigger juga bisa dengan mudah mengatur ukuran minimal dan maksimal atribut tersebut. Jika atribut yang dibuat ternyata tidak diperlukan, *rigger* bisa dengan mudah menghapusnya dengan fitur *Delete Attribute*. Pada fitur tersebut, *rigger* bisa memilih atribut mana yang ingin dihapus.

## 2.11 Menghubungkan *Node* Atribut

Allen & Murdock (2008) menjelaskan salah satu fitur yang sangat berguna pada *Attribute Editor* terlebih untuk proses rigging ialah atribut bisa dihubungkan satu sama lain, baik itu dalam satu objek, maupun dalam objek – objek yang berbeda. Atribut – atribut tersebut dapat dihubungkan dengan fitur *Set – Driven Keys, Connection Editor,* dan *Expression Editor.*

### 2.11.1. Connection Editor

Dengan *Connection Editor,* rigger bisa dengan mudah menghubungkan satu atribut dengan atribut lainnya. Contohnya, jika atribut *translate X* dihubungkan dengan atribut *rotate X*, maka saat objek dipindahkan dalam sumbu X sejauh 10, objek tersebut juga akan berputar sebanyak 10 derajat. Setelah dihubungkan, Kotak atribut yang telah dihubungkan dengan *Connection Editor* akan berubah menjadi kuning.



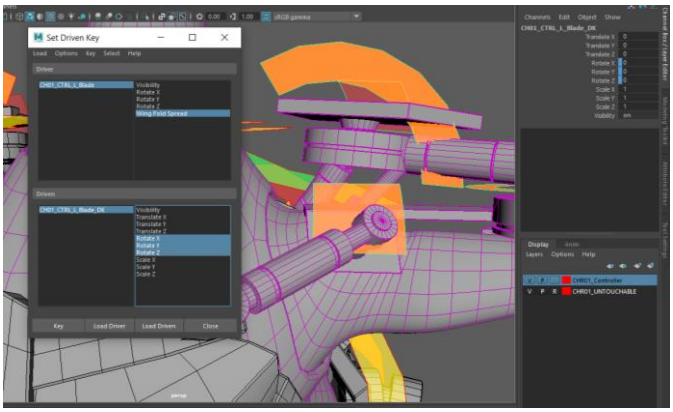
Gambar 2.9. *Connection Editor*

(dokumentasi pribadi)

### 2.11.2. Set – Driven Keys

Berfungsi sama seperti *Connection Editor*. Hanya saja dengan *Set – Driven Keys*, *rigger* bisa mengatur hingga pada ukuran berapa atribut tersebut dihubungkan. Contohnya, jika pada ukuran 0 atribut *translate* X objek dihubungkan dengan atribut *rotate* X, kemudian pada ukuran 10 atribut *translate* X dihubungkan dengan atribut *rotate* X yang berukuran 15. Maka saat objek dipindah dalam sumbu X hingga ukuran atribut *translate* Xnya 10, maka objek tersebut juga akan berputar pada sumbu X sebanyak 15 derajat. Namun, jika objek tersebut dipindahkan pada

sumbu X lebih jauh, atribut *rotate* X tidak akan berubah. Saat dihubungkan, kotak atribut yang dihubngkan akan berubah warna menjadi biru.

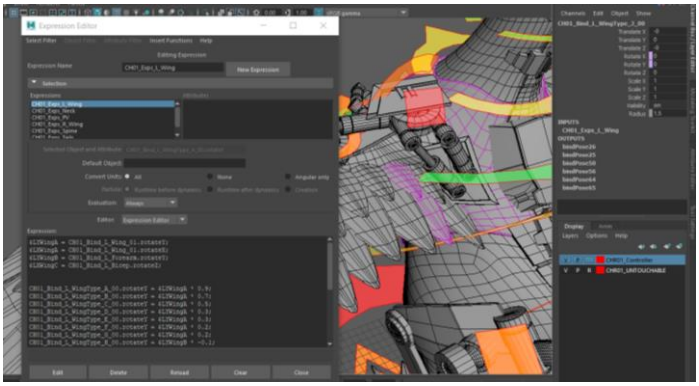


Gambar 2.10. *Set driven key*

(dokumentasi pribadi)

### 2.11.3. Expression Editor

Beane (2012) menjelaskan bahwa Expression editor memiliki fungsi yang mirip dengan *Connection Editor* dan *Set – Driven Keys*. Namun, *Expression Editor* dijalankan dengan *scripting*. Bahasa pemrograman yang digunakan untuk *Expression Editor* adalah bahasa MEL (*Maya Embended Language*).



Gambar 2.11. *Expression Editor*

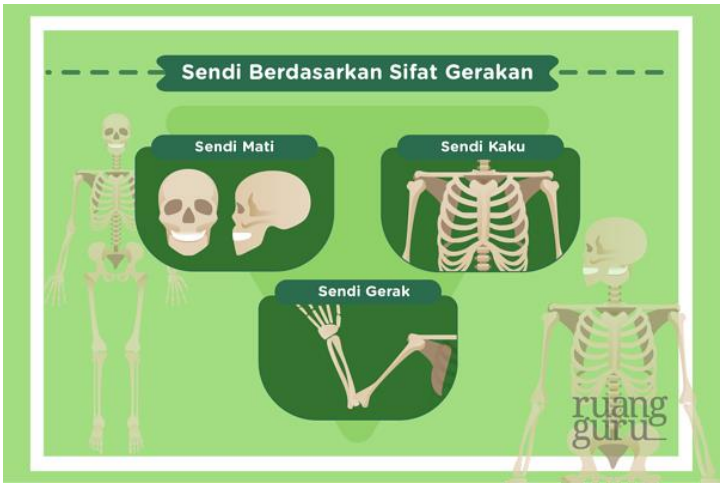
(dokumentasi pribadi)

## 2.12 Tulang

Tulang adalah kerangka yang berfungsi sebagai penopang tubuh makhluk vertebrata, seperti mamalia, beberapa jenis ikan, dan unggas. Hal ini diperkuat oleh Diniari (2018) yang menjelaskan bahwa tulang merupakan alat gerak pasif tubuh. Samiadi (2017) menambahkan bahwa tulang merupakan bagian tubuh yang sangat keras yang melindungi organ – organ tubuh. North dan Bell (1990) juga menambahkan bahwa tulang – tulang yang ada dalam tubuh dihubungkan dengan sendi – sendi dan otot yang melekat padanya. Dengan demikian jika tidak ada tulang tubuh tidak akan bisa berdiri tegak.

## 2.13 Sendi

Kresnoadi (2018) menjelaskan bahwa sendi adalah bagian dari kerangka makhluk vertebrata yang berbentuk menyerupai bantalan yang menghubungkan tulang – tulangnya. Tanpa sendi, tulang – tulang hanya akan diam mengambang di dalam tubuh dengan otot - otot melekat padanya. Sendi juga berfungsi mengatur bagaimana tulang – tulang yang terhubung berartikulasi. Berdasarkan sifat gerakannya sendi dibagi menjadi tiga, yaitu sendi mati, sendi kaku, dan sendi gerak.




Gambar 2.12. Sendi berdasarkan sifat gerakan

(https://blog.ruangguru.com/macam-macam-sendi)

1. **Sendi mati**

   Sesuai dengan namanya, sendi mati merupakan sendi yang tidak bisa bergerak sama sekali. Salah satunya terletak pada tulang tengkorak.

2. **Sendi kaku**

   Sendi ini merupakan sendi yang memiliki gerakan – gerakan terbatas. Contohnya yaitu sendi yang terdapat pada tulang belakang dan tulang rusuk.

3. **Sendi gerak**

   Sendi ini merupakan sendi yang bisa bergerak dengan leluasa. Sendi ini umumnya terdapat pada tulang – tulang yang melakukan gerakan aktif, seperti pada tulang lengan, tulang bahu, tulang pergelangan tangan dan lain – lainnya.

Berdasarkan arah geraknya sendi dibedakan menjadi enam, yaitu sendi geser, sendi peluru, sendi gulung, sendi engsel, sendi putar, dan sendi pelana.


Gambar 2.13. Sedni berdasarkan arah pergerakan

(https://blog.ruangguru.com/macam-macam-sendi)

1. **Sendi geser (*plane*)**

   Sendi ini merupakan sendi yang memungkinkan tulang bergeser dengan tulang lainnya, seperti pada ruas tulang belakang.

2. **Sendi engsel (*hinge*)**

   Sendi ini merupakan sendi yang memungkinkan tulang bisa berputar pada satu arah saja, seperti pada tulang siku dan lutut.

3. **Sendi gulung (*condylar*)**

   Sendi ini merupakan sendi yang memungkinkan tulang berputar pada porosnya dengan gerakan yang terbatas, seperti pada tulang hasta dan pengumpil.

4. **Sendi Putar (*pivot*)**

   Sendi ini merupakan sendi yang memungkinkan tulang berputar pada poros ujung tulang lainnya, seperti pada tulang tengkorak dan atlas.

5. **Sendi peluru (*ball and socket*)**

   Sendi ini merupakan sendi yang memungkinkan tulang berputar ke segala arah, seperti pada tulang lengan atas dan gelang bahu.

6. **Sendi pelana (*saddle*)**

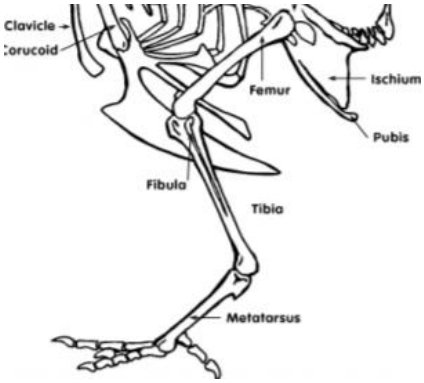
   Sendi ini merupakan sendi yang memungkinkan tulang berputar ke dua arah, yaitu arah samping dan depan, seperti pada tulang pangkal ibu jari.

## 2.14 Kerangka Tubuh Ayam

Ayam merupakan hewan *vertebrata* yang memiliki kerangka tulang tertanam pada badannya. Kerangka ini meliputi kerangka tulang kaki, sayap, badan, kepala, dan kerangka ekor.

### 2.14.1. Kerangka Kaki Ayam

Pada kaki ayam terdapat tulang Femur atau tulang paha yang pada bagian bawahnya berartikulasi dengan tulang *tibia* (tulang kering) dan bagian luar tulang *fibula* (tulang betis). King dan McLelland (1975) menambahkan bahwa tulang *femur* hanya bergerak ke arah depan dengan sendi engsel. Bagian bawah tulang *tibia* berartikulasi dengan tulang *tarmetasaurus*. Tarmetasaurus sendiri merupakan gabungan dari ruas – ruas bagian bawah tulang *tarsal* (tulang telapak kaki) dan tulang *metasaurus*.

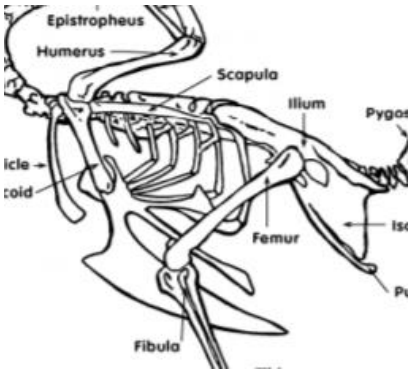



Gambar 2.14. Kerangka kaki ayam

(https://www.pintarbiologi.com/2016/02/sistem-rangka-pada-ternak-unggas.html)

### 2.14.2. Kerangka Badan Ayam

McLelland (1990) menjelaskan bahwa tulang punggung pada ayam terdiri dari *notarium* yang merupakan tulang utama dari tulang punggung dan membentuk *cervical vertebrae*. Budi (2012) menambahkan bahwa pada tulang belakang ayam, terdapat 8 ruas tulang belakang.

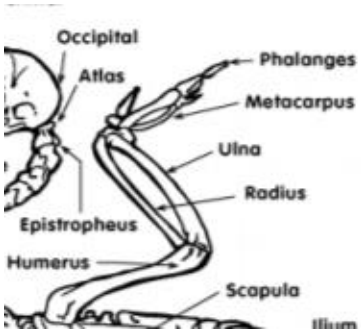



Gambar 2.15. Kerangka badan ayam

(https://www.pintarbiologi.com/2016/02/sistem-rangka-pada-ternak-unggas.html)

### 2.14.3. Kerangka Sayap Ayam

Pada kerangka sayap ayam terdapat tulang *humerus*, tulang bahu depan yang terdiri dari *radius* dan *ulna*, dan tulang *carpus*. Nickel et al. (1977) menambahkan bahwa tulang *humerus* bergerak berputar dengan menggunakan sendi putar, tulang radius berartikulasi dengan sendi engsel.

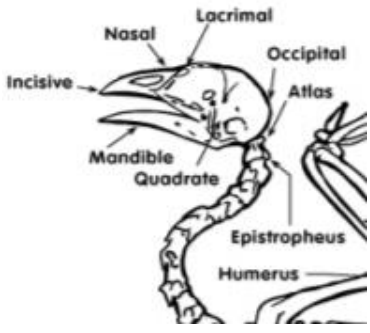



Gambar 2.16. Kerangka sayap ayam

(https://www.pintarbiologi.com/2016/02/sistem-rangka-pada-ternak-unggas.html)

### 2.14.4. Kerangka Kepala Ayam

Kepala ayam terdiri dari tulang tengkorak, tulang *maxilla*. Sandi (2011) menambahkan bahwa, pada leher ayam terdapat 13 hingga 14 ruas tulang.

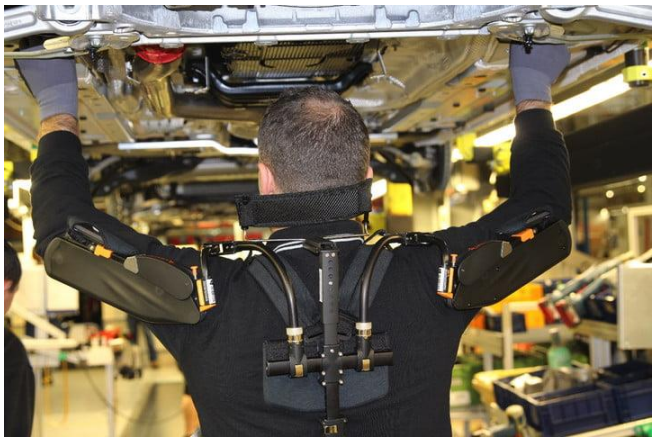



Gambar 2.17. Kerangka kepala ayam

(https://www.pintarbiologi.com/2016/02/sistem-rangka-pada-ternak-unggas.html)

## 2.15 Exosuit

Levitate Technologies (2018) menjelaskan bahwa *exosuit* merupakan kumpulan *frame* yang dirakit dan dipasang ke tubuh manusia. *Frame* tersebut bisa berupa logam seperti besi ataupun bahan yang lembut seperti kain atau fabrik khusus. Ferguson (2018), McGowan (2019), dan Li (2018) menambahkan bahwa *frame exosuit* dirakit dengan susunan komponen elektrik, penumatik, atau hidrolik sehingga dapat memberikan tenaga atau kekuatan tambahan.



Gambar 2.18. *Exosuit*

(https://www.digitaltrends.com/cool-tech/fire-resistant-levitate-airframe-exosuit/)

Levitate Technologies (2018) menjelaskan bahwa, *frame exosuit* memiliki sensor yang mendeteksi gerakan anggota tubuh penggunanya. Dari gerakan yang dideteksi tersebut, sensor akan mengalirkan informasi semua beban kepada *core exosuit* tersebut, lalu menggerakkan *frame* sesuai beban yang dialirkan. Dengan demikian pekerjaan berat akan terasa lebih ringan bagi penggunanya.