# BAB III
# PELAKSANAAN KERJA MAGANG

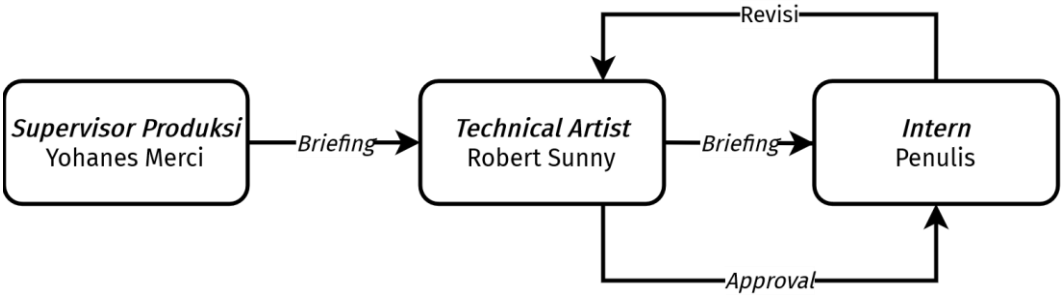
## 3.1. Kedudukan dan Koordinasi

### 1. Kedudukan

Penulis bekerja sebagai *Intern* di bawah divisi *Technical Artist* di UMN Pictures. Divisi tersebut dipimpin oleh *Lead Technical Artist*, Bapak Robert Sunny, dan disupervisi oleh Bapak Andrew Willis selaku *supervisor* teknis.

### 2. Koordinasi

Selama penulis melakukan kerja magang, alur koordinasi yang dilakukan mempunyai struktur sebagai berikut. Bapak Yohanes Merci, selaku *supervisor* produksi, mendapat informasi dari para *trainee* tentang *workflow* yang lambat dan belum rapi. Dari sana, Bapak Robert diberikan tugas untuk membuat *tools* untuk membantu para *trainee* mengerjakan tugas mereka. Terakhir, tugas tersebut dibagi dengan penulis selaku *intern technical artist*. Pada saat pengerjaan, penulis meminta *approval* dan memberikan revisi ke Bapak Robert.




Gambar 3.1. Bagan Koordinasi Pembuatan *Training Tools*

Dokumentasi Pribadi

## 3.2. Tugas yang Dilakukan

Berikut adalah tugas yang dikerjakan oleh penulis saat magang di UMN Pictures:

Tabel 3.1. Detail Pekerjaan Yang Dilakukan Selama Magang

Dokumentasi Pribadi

| No. | Minggu | Proyek | Keterangan |
|---|---|---|---|
| 1. | 18-21 Februari 2020 | Wildeast | • Mempelajari Git *version control system*<br>• Mempelajari *framework* Jasmine<br>• Membuat sketsa UI<br>• Membuat *function* utama |
| 2. | 24-28 Februari 2020 | | • Membuat UI<br>• Implementasi UI<br>• Penambahan fitur *lock*<br>• *Bugfix* |
| 3. | 2-6 Maret 2020 | | • Revisi UI<br>• Penambahan fitur *global*<br>• *Bugfix* |
| 4. | 9-13 Maret 2020 | *Training* Animasi 2020 | • Menyiapkan *repository*<br>• Integrasi *quick playblast* ke *training tools*<br>• Membuat sketsa UI untuk *scene setup*<br>• Mempelajari GraphQL |
| 5. | 16-20 Maret 2020 | | • Membuat UI untuk *scene setup*<br>• Implementasi UI<br>• Menyelesaikan fitur *filtering* untuk *scene setup* |

| | | | |
|---|---|---|---|
| | | | • Membuat *function* untuk *save local*<br>• Revisi UI untuk *scene setup*<br>• *Bugfix* |
| 6. | 23-27 Maret 2020 | | • Mempelajari RegEx<br>• Menambah fitur *versioning* ke *save new version*<br>• *Bugfix* untuk *quick playblast*<br>• Revisi UI *scene setup* |
| 7. | 30 Maret - 3 April 2020 | Ilo | • Mempelajari gFur dan solusi lainnya untuk *fur* |
| 8. | 6-10 April 2020 | Candy Monster | • Porting model ke *project* Unreal Engine 4<br>• Setup *material* |
| 9. | 13-15 April 2020 | | • Porting model ke *project* Unreal Engine 4<br>• Setup *material* |

### 3.3. Uraian Pelaksanaan Kerja Magang

Penulis mengembangkan dua *tool* selama periode magang di UMN Pictures. Pertama adalah *render manager* yang dikembangkan selama tiga minggu pertama. Kedua adalah *training tools* yang dikerjakan tiga minggu berikutnya. Kedua *tools* tersebut digunakan untuk mempermudah pekerjaan dalam *software* Maya. Pengembangan dilakukan dalam bahasa pemrograman Python dengan *software* Visual Studio Code, dikelola menggunakan *version control system* Git, dan di-*upload* ke *online repository* Bitbucket. Untuk tiga minggu terakhir, penulis membantu proses *research and development* dari tim *render* untuk mempelajari

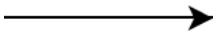
tentang *render engine* baru. Dalam hal ini, penulis ditugaskan untuk mempelajari Unreal Engine 4.

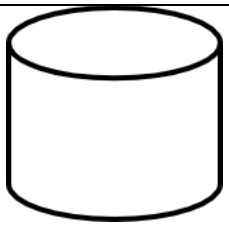
Untuk laporan ini, proses pelaksanaan yang akan dibahas adalah pengembangan *training tools*.

Di bawah akan disertakan *flowchart* yang menggambarkan *workflow* dari masing-masing *tool*. Oleh karena itu, sebagai referensi, berikut adalah tabel definisi simbol-simbol yang digunakan:

Tabel 3.2. Referensi simbol *flowchart* dan definisinya

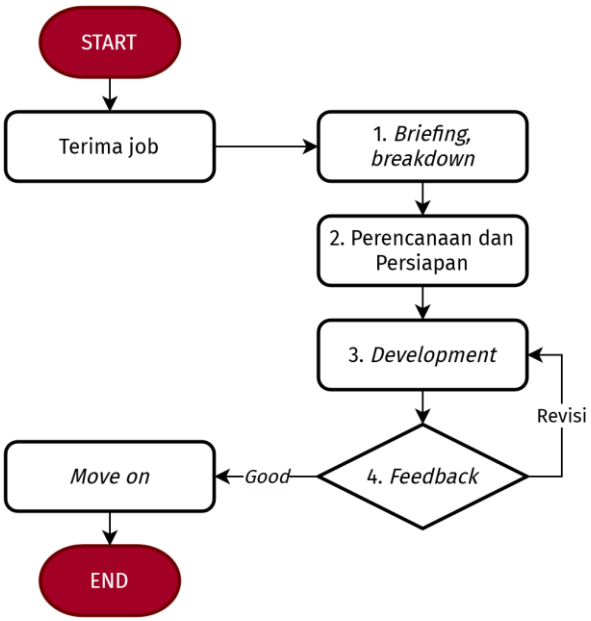
Dokumentasi Pribadi

| Simbol | Definisi |
|---|---|
|  | ***Terminator***, menandakan awal dan akhir dari sebuah program. |
|  | ***Process***, menandakan sebuah tahap pekerjaan yang dilakukan oleh program. |
|  | ***Flow***, menandakan alur dari eksekusi program. |
|  | ***Predefined process***, menandakan sebuah *process* yang didefinisikan di bagian lain. Penulis menggunakannya untuk enkapsulasi sebuah *tool*. |
|  | ***Document***, menggambarkan sebuah *file*. |

| | |
|---|---|
|  | ***Computer***, penulis menggunakannya untuk menggambarkan PC yang digunakan oleh *trainee* atau *admin*. |
|  | ***Database***, penulis menggunakannya untuk menggambarkan *server* yang digunakan oleh studio. |

### 3.3.1. Proses Pelaksanaan

Ketika divisi *technical artist* mendapat *job* untuk membuat *tool*, proses yang dilakukan dapat dibagi menjadi empat tahap utama. Berikut adalah *flowchart* yang menggambarkan tahapan tersebut.




Gambar 3.2. *Flowchart* pengerjaan *tools*

Dokumentasi Pribadi

### *1. Briefing*

Pada tanggal 09 Maret 2020 hingga 15 April 2020, UMN Pictures mengadakan *training* animasi untuk karyawan. Program *training* ini tidak hanya diperuntukkan kepada *animator* saja, tetapi karyawan dari divisi manapun dapat mengikutinya. Penulis tidak mengikuti *training* ini karena penulis mendapat tugas dari Bapak Robert untuk membantu mengerjakan *tools* untuk *training* tersebut.

Bapak Robert mendapat permintaan dari Bapak Merci untuk mengembangkan *tool* untuk membantu proses *training*. *Tool* yang akan dibuat ini mengakomodasi proses *trainee management* (*job assignment*, penilaian dari *supervisor*), *quick playblast*, *scene setup*, *save file management. Tools* tersebut tidak hanya akan dipakai untuk *training* berikut, tetapi juga akan digunakan untuk *training-training* di masa depan.

### 2. Perencanaan dan Persiapan

*Workflow* yang akan dikerjakan oleh para *trainee* di-*breakdown* terlebih dahulu oleh Bapak Robert. Tujuannya adalah untuk mengetahui secara konkret pekerjaan yang harus dilakukan. Berikut adalah tahapannya:

1) Pertama, *supervisor* dan *manager training* melakukan *briefing* untuk para *trainee* tentang tugas yang akan diberikan pada minggu tersebut. Para *trainee* lalu dapat memilih untuk mengerjakan tugas dalam *list job*-nya, baik itu tugas untuk minggu tersebut ataupun *retake*/revisi dari tugas minggu lalu.
2) Setelah *trainee* memilih tugas yang akan dikerjakannya, tahap kedua yaitu, *scene* Maya disiapkan. Persiapan ini mempunyai tiga tahap. Pertama, *trainee* mengisi informasi *shot* di *node* ‘shotInfo’. Informasi ini akan digunakan untuk menciptakan nama *file* yang unik. Kedua adalah menyiapkan *camera* beserta *setting* yang sesuai dengan permintaan *shot* tersebut. Terakhir adalah membuat *reference* ke *rig* karakter, akan tetapi untuk tahap ini akan dilakukan secara manual.

3) Ketika *trainee* sedang mengerjakan animasi, tahap ketiga yaitu, '*Save as*' di tiga tempat. Detailnya akan dijelaskan di bawah.
4) Apabila animasi telah selesai, tahap keempat yaitu, membuat *playblast* dari animasi yang telah dibuat.
5) Terakhir, tahap kelima, *supervisor* mengecek hasil kerja dari *trainee* dan memutuskan apakah *shot* tersebut diterima (*approved*), revisi, atau *retake*.

Setelah melalui *briefing* singkat dengan Bapak Robert, penulis diberi tanggung jawab untuk membuat *tools* untuk mengakomodasi *workflow* tahap 2, 3, dan 4. *Tools* tersebut diberi nama '*scene setup'*, '*save file'*, dan '*quick playblast'*. Bapak Robert akan menyiapkan *server*, *website*, dan *tool 'upload to server'* untuk mengakomodasi tahap 1 dan 5. Bapak Robert juga bertanggung jawab untuk menggabungkan seluruh *code* tersebut dan mengintegrasikannya ke Maya.

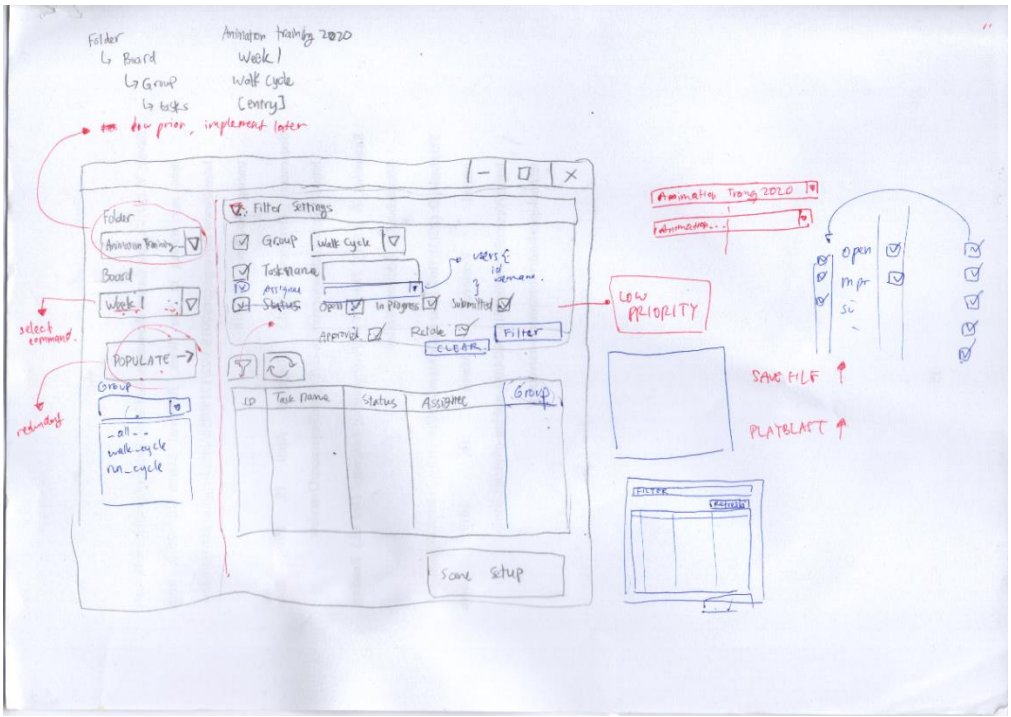
'*Scene setup'* dan '*quick playblast'* telah dibuat sebelumnya tetapi *tools* tersebut terintegrasi dengan Jasmine *assist*. Jasmine *assist* adalah *tools framework* yang dibuat oleh Bapak Robert khusus untuk UMN Pictures. Jasmine *assist* terdiri dari berbagai macam *tools* yang membantu *pipeline* produksi studio. Pada *training* ini, *trainee* tidak menggunakan Jasmine *assist*, sehingga tim *technical artist* harus menyediakan yang baru.

Oleh karena *tools* ini terpisah dari Jasmine, Bapak Robert membuat *repository* baru di Bitbucket dengan nama 'umnp-training-tools'. Bitbucket dipilih karena pada pengembangan Jasmine digunakan Bitbucket, sehingga Bapak Robert telah berpengalaman menggunakannya. Penulis diminta untuk menyiapkan *repository* tersebut agar dapat digunakan sebagai *environment* untuk bekerja.

Agar tidak memakan waktu yang banyak untuk membuat sistem yang baru, penulis mencoba untuk meniru apa yang telah dikerjakan di Jasmine. *Path* ke *folder-folder* penting ditempatkan dalam sebuah *file* bernama 'settings.py', lalu setiap *tool* ditempatkan dalam folder yang diberi *file* kosong bernama

‘__init__.py’ untuk memberi tanda ke Python bahwa *folder* tersebut adalah sebuah *package*, sehingga dapat digunakan untuk bekerja. Penulis juga menerapkan *naming convention* berupa awalan ‘tr_’ sebagai pembeda bahwa *file* tersebut milik *training tools*. Contoh : ‘tr_settings.py’. Ini dilakukan untuk menghindari kesamaan dengan Jasmine.

Sebelum memulai pengembangan ‘*scene setup*’, penulis membuat sketsa UI (*User Interface*) di kertas. Sketsa ini diberikan revisi dan catatan oleh Bapak Robert.




Gambar 3.3. Sketsa dan revisi untuk ‘*scene setup’*

Dokumentasi Pribadi

## 3. Proses Pengembangan

Penggunaan dari Git *version control system* telah dipelajari oleh penulis pada saat penulis mengembangkan ‘*render manager’* pada minggu 1-3 magang. Oleh karena itu, penulis mulai terbiasa dengan cara bekerja secara berkolaborasi dengan Bapak Robert.

Sebagai pengenalan singkat, Git berfungsi dengan cara menyimpan '*snapshot*' dari proyek yang sedang dikerjakan setiap kali perintah '*commit*' dijalankan. '*Snapshot*' tersebut menjadi bagian dari sejarah proyek tersebut sebagai versi baru. Pada pengerjaan sebuah proyek, akan terdapat banyak *commit*, yang bisa jadi stabil atau malah merusak. Oleh karena itu, dengan penggunaan git, proyek dapat di-*rollback* ke versi manapun selama tercatat dalam sejarah. Oleh karena itu, git disebut dengan '*version control system*'.

Tentu saja, dalam pengerjaan kebanyakan proyek, akan terdapat banyak kontributor (dalam kasus ini, Bapak Robert dan penulis adalah dua kontributor tersebut). Ini berarti, dalam waktu yang bersamaan, bisa jadi para kontributor mengerjakan hal yang sama atau berbeda, sehingga sulit untuk mengkoordinasikannya. Oleh karena itu, git mempunyai perintah '*branch*' dan '*merge*'. Dengan perintah '*branch*', sejarah dari proyek dapat bercabang, membentuk *timeline* yang baru. Dasar dari pembagian cabang ditetapkan oleh *management* masing-masing proyek. Tetapi yang menjadi kesamaan adalah *branch* 'master' sebagai batang utama dari sebuah proyek. Ketika sebuah *branch* selesai dikerjakan, dapat dilakukan perintah '*merge*' yang bertujuan untuk menggabungkan *branch* menjadi satu *timeline*, selayaknya tidak ada percabangan sama sekali.

Gambar 3.4. Visualisasi dari *timeline* sebuah proyek. Biru adalah *branch* 'master'

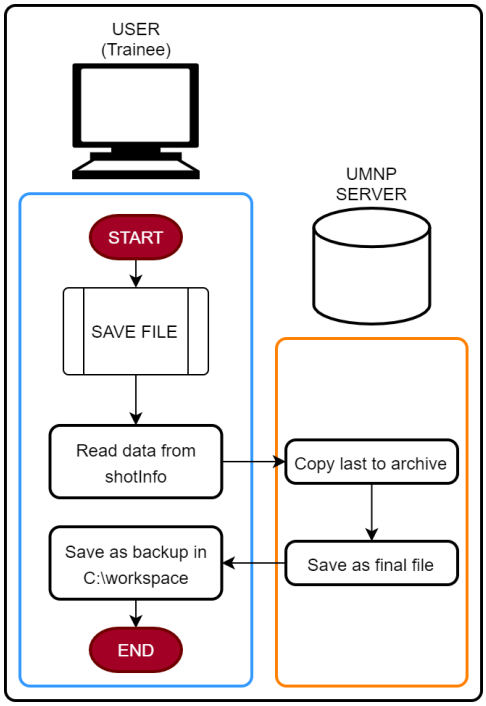
Dokumentasi Pribadi

Bitbucket di sini berfungsi sebagai *remote repository*. '*Repository*' adalah istilah untuk tempat penyimpanan *software*, sedangkan '*remote*' berarti *repository* yang sama yang digunakan oleh semua kontributor, berbeda dengan '*local*' yang berarti digunakan secara lokal oleh masing-masing kontributor. '*Remote'* juga sering disebut bergantian dengan '*origin*'.

Dalam berinteraksi dengan *repository*, perintah yang paling sering digunakan adalah '*push*' dan '*pull*'. '*Push*' digunakan untuk mengunggah semua *commit* yang dilakukan di *local repository* ke *remote*. '*Pull*' digunakan untuk mengunduh semua perubahan yang terjadi di *remote* untuk disimpan di *local*.

Seperti yang telah disebutkan di atas, tergantung dari *management* tim untuk menggunakan *workflow* yang tepat dengan menggunakan fitur-fitur dalam git. Untuk pengerjaan *training tools*, tidak boleh dikerjakan secara langsung dalam *branch* 'master'. Apa yang terdapat dalam *branch* 'master' harus bersifat siap rilis. Penulis dan Bapak Robert akan bekerja di *branch* baru. *Progress* di *branch* tersebut hanya akan di-*merge* ke *branch* 'master' apabila fitur yang dikerjakan telah selesai atau siap untuk digunakan.

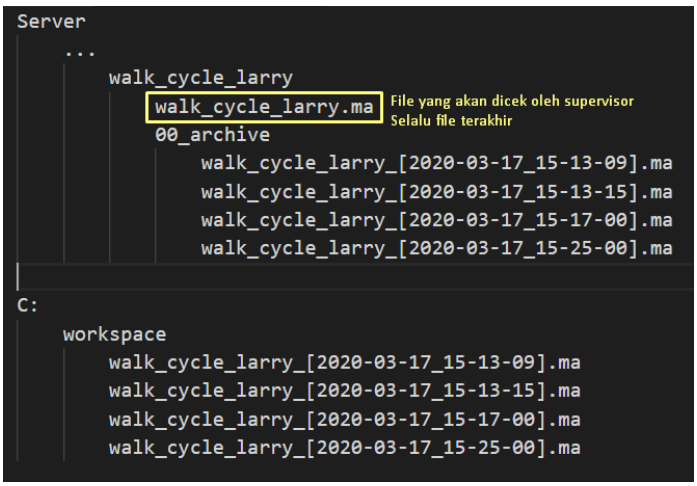
Fitur pertama yang dikerjakan adalah '*save file*' karena yang paling sederhana dibanding fitur yang lain. Seperti yang telah dijelaskan di tahap *breakdown*, *tool* ini akan melakukan tiga kali *save*. Dengan menggunakan data yang terdapat di *node* 'shotInfo', *tool* ini menentukan letak *save file* yang benar di *server* dan nama *file* yang sesuai dengan *convention*. Otomatisasi proses ini dilakukan untuk menghindari *human error*. Pertama, *tool* meng-*copy file* terakhir ke folder '00_archive' apabila sudah terdapat *save file*. *File archive* tersebut lalu diberi *timestamp* agar penamaannya unik. Kedua, *tool* melakukan *file save* di *file final* yang nantinya akan dicek oleh *supervisor*. Terakhir, *save file* di 'C:\workspace' sebagai *backup*.




Gambar 3.5. *Workflow* dari '*save file*'

Dokumentasi Pribadi

Penggunaan *prefix* angka seperti '00_archive' disarankan oleh Bapak Robert karena windows akan mengurutkan *folder* berdasarkan angka tersebut. Urutan tersebut bisa disesuaikan dengan *workflow*.




Gambar 3.6. Ilustrasi isi *folder* yang di-*manage* oleh *tool* ini

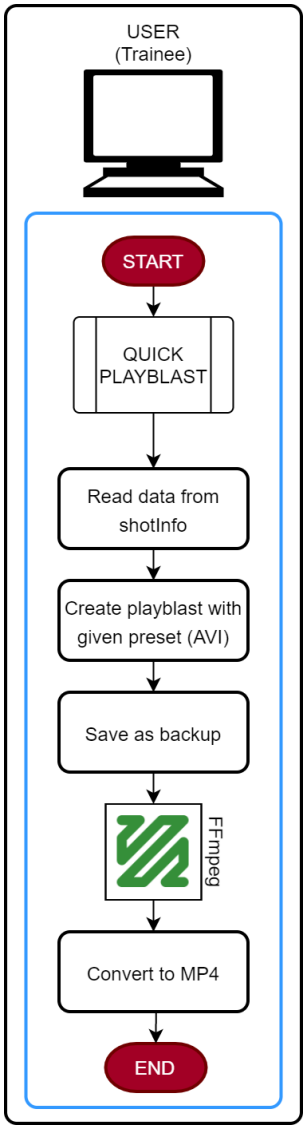
Dokumentasi Pribadi

Fitur yang dikerjakan selanjutnya adalah '*quick playblast*'. Menurut dokumentasi Maya 2017, '*Playblast*' adalah metode untuk membuat 'sketsa' animasi yang cepat, kasar, tetapi menggambarkan hasil akhir dari animasi yang dikerjakan. Tujuan dari *playblast* adalah memperoleh *preview* sebelum proses *render*, sehingga dapat di-*review* tanpa harus menunggu *render*. Pembuatan *playblast* dalam Maya membutuhkan pengguna mengatur banyak opsi yang dapat disesuaikan dengan kebutuhan. Dalam *training* animasi ini, kebutuhan *playblast* ditentukan oleh *supervisor*. Ini membuat proses pembuatan *playblast* menjadi lambat dan rawan dengan *human error*. Oleh karena itu, '*quick playblast*' dibuat untuk mempercepat dan men-*streamline* proses *playblast* sehingga hasil *playblast* seragam antar semua *trainee*.

'*Quick playblast*' ini pernah dibuat di Jasmine assist, Bapak Robert menyarankan untuk mengambil dari *code* Jasmine dan diintegrasi ke *training tools* sesuai kebutuhan. Oleh karena itu *branch* pertama dinamai 'playblastSaveFile'.

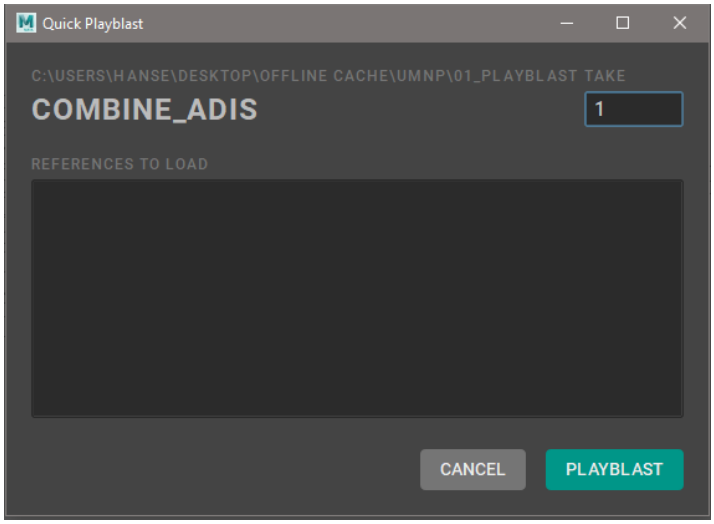
Spesifikasi yang diminta oleh *supervisor* adalah : resolusi 1280x720, format MOV H264, diberi nama *shot* dan nama *trainee*, dan nomor *frame* di bawah kanan. Nomor *take* dari *playblast* akan di-*generate* secara otomatis, tetapi pengguna dapat memasukkan sendiri nomor *take* yang akan digunakan. Selain membuat *playblast*, *tool* ini juga membuat *save file* baru sebagai *backup*. Kedua *file* tersebut disimpan di folder '01_playblast'.

*Playblast* yang dibuat oleh Maya berupa *file* AVI, sedangkan yang diminta adalah file MOV. Sebelumnya Bapak Robert menggunakan *tool* bernama 'ffmpeg' untuk melakukan konversi ke MP4. Sesuai arahan Bapak Robert, penulis menggunakan *tool* yang sama tetapi mengkonversi ke MOV. *File* AVI yang dibuat juga akan dihapus oleh '*quick playblast*' setelah konversi selesai.




Gambar 3.7. *Flowchart* dari '*quick playblast*'

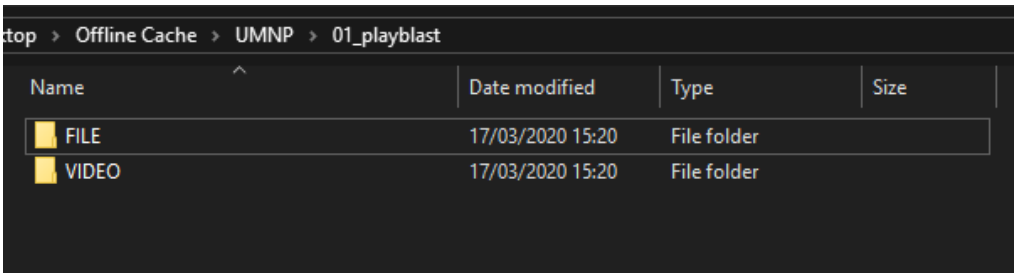
Dokumentasi Pribadi




Gambar 3.8. *Interface* dari '*quick playblast'*

Dokumentasi Pribadi

Sayangnya, '*quick playblast*' untuk Jasmine tidak mengimplementasikan fitur *save file*. Bapak Robert meminta penulis untuk mengimplementasikannya. Penulis lalu mengimplementasikannya dengan menggunakan fitur '*save file*' yang telah dibuat sebelumnya. *File* Maya tersebut disimpan dalam *folder* 'FILE', sedangkan hasil *playblast* disimpan dalam folder 'VIDEO'.

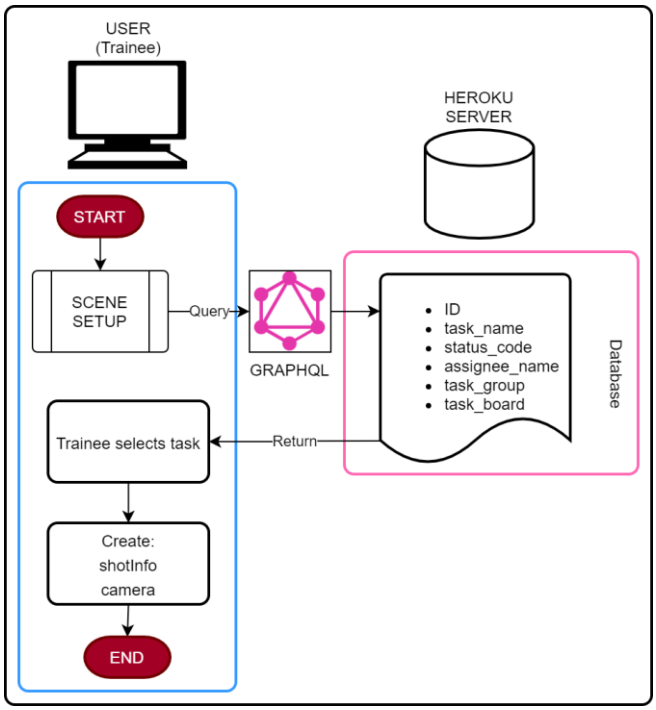



Gambar 3.9. *Folder* yang dibuat oleh '*quick playblast*'

Dokumentasi Pribadi

*Tool* selanjutnya yang dikerjakan adalah '*scene setup*'. *Tool* ini digunakan untuk mempersiapkan *scene* sebelum *trainee* memasukkan *reference* dan bekerja di dalamnya. Proses persiapan tersebut meliputi: membuat *node* 'shotInfo', dan membuat *camera*. *Trainee* akan memilih *board* dan *group* untuk

ditampilkan dalam sebuah tabel. Apabila *trainee* kesulitan untuk menemukan pekerjaannya, maka *filter* yang terdapat di '*Filter settings*' dapat digunakan. Setelah pengguna memilih pekerjaan yang akan dikerjakan, maka tombol *scene setup* akan melakukan *setup* secara otomatis.



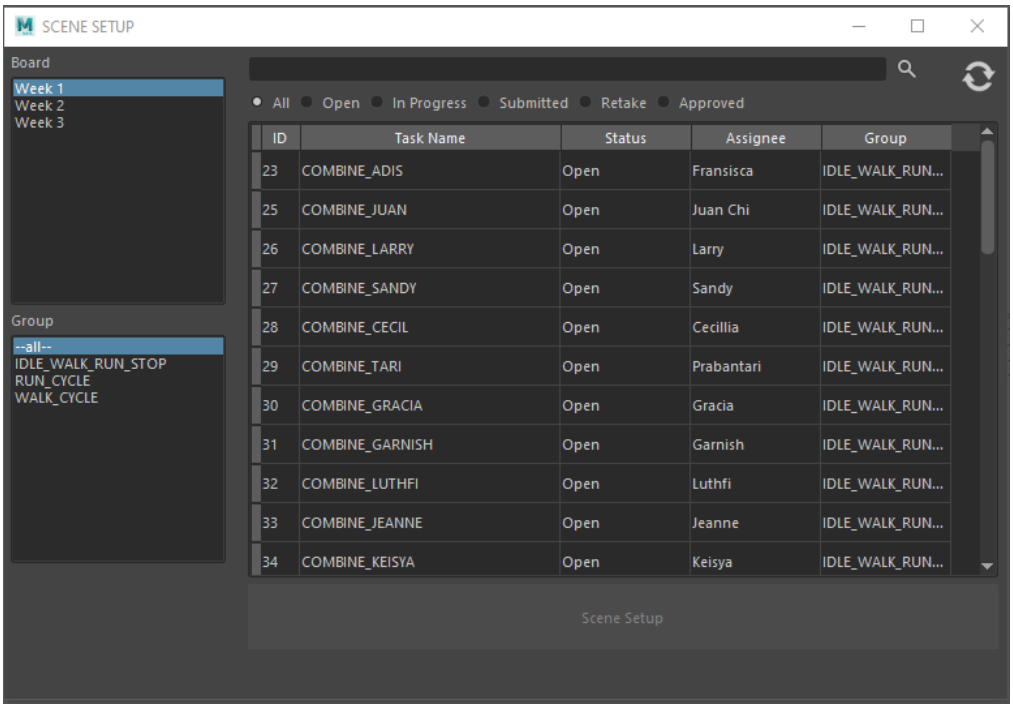

Gambar 3.10. *Flowchart* dari '*scene setup*'

Dokumentasi Pribadi

Informasi dari *shot* beserta *status*-nya didapatkan dari *website* yang disiapkan oleh Bapak Robert. Di *website* tersebut terdapat *database* yang bisa di-*query* menggunakan GraphQL. '*Query*' memiliki arti yaitu menanyakan sesuatu. Dalam hal ini, '*query*' berarti meminta data kepada *database* tersebut. GraphQL adalah sebuah *Query Language* yang bertujuan untuk berkomunikasi dengan *database*. GraphQL dapat dengan mudah diimplementasikan dengan Python, oleh karena itu Bapak Robert memilih bahasa ini. Penulis belum pernah menggunakan GraphQL sebelumnya, oleh karena itu, penulis mempelajari penggunaannya tentunya dengan arahan Bapak Robert.

Seperti *database* pada umumnya, informasi yang di-*query* menggunakan GraphQL adalah berupa *dictionary*. *Dictionary* dalam bahasa pemrograman Python adalah tipe data yang berisi *key* dan *value*. Informasi yang dibutuhkan berada di *value*. Untuk mendapatkan informasi yang tepat, *key* yang benar harus digunakan. Isi dari *value* tersebut yang akan digunakan untuk mengisi tabel *shot* dalam '*scene setup*'.

Dari revisi yang diberikan oleh Bapak Robert, akhirnya penulis mengubah UI-nya hingga seperti gambar di bawah. '*Filter settings*' dapat disederhanakan hingga hanya *search bar* dan *radio button* untuk *status*. Pemilihan *board* dan *group* menggunakan *text scroll list* karena dapat menghemat jumlah klik. Tombol '*Populate*' yang digunakan untuk mengisi tabel dihilangkan, fungsinya dilakukan ketika *trainee* telah selesai memilih *board* dan *group* yang ingin ditampilkan. Ini juga dapat menghemat jumlah klik.




Gambar 3.11. '*Scene Setup'* pada tahap *final*.

Dokumentasi Pribadi

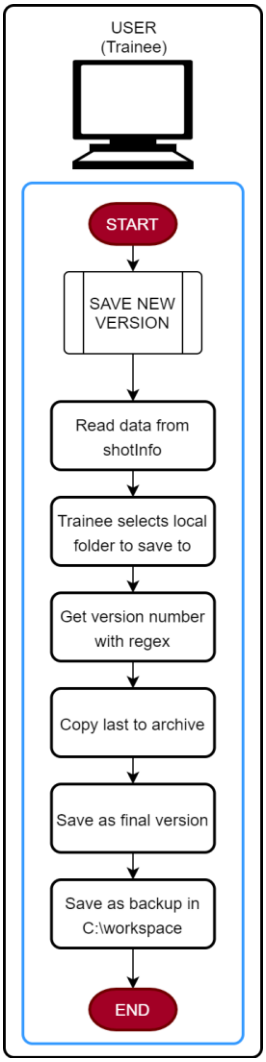
Mulai tanggal 16 Maret, UMN Pictures tidak melanjutkan pekerjaan di studio. Karyawan bekerja dari rumah (*Work From Home*) oleh karena pandemik Covid-19. Keadaan tersebut memaksa *trainee*, termasuk tim *technical artist*, untuk menggunakan komputer / laptop pribadi di rumah masing-masing. Kebutuhan *tools* berubah drastis karena komputer *trainee* tidak lagi terhubung dengan *server* secara langsung. Berikut adalah poin-poin perubahan:

- Perlu dibuat sistem *upload* agar pekerjaan yang dilakukan *trainee* dapat sampai di *server* UMNP.
- '*Save file*' juga tidak lagi dapat menyimpan di *server* sehingga harus di-*reroute*.
- *Tools* sebelumnya disimpan di *server* sehingga mudah diakses oleh pengguna dan mudah di-*manage* oleh *technical artist*. Karena tidak adanya akses langsung, *tools* perlu didistribusikan ke komputer *trainee*.

Bapak Robert mengerjakan masalah *upload*. Penulis ditugaskan untuk mengubah '*save file*' dan '*quick playblast*'.

Bersamaan dengan perubahan tersebut, berdasarkan diskusi bersama Bapak Robert, '*save file*' menggunakan *timestamp* sulit dibaca. Oleh karena itu diputuskan untuk menggunakan nomor versi saja. '*Save file*' berubah menjadi '*save new version*'.

Penulis melakukan *rerouting* semua *link* yang digunakan dalam '*save new version*' sehingga tidak lagi mengarahkan ke *server*. *Save* dilakukan ke *folder* yang dipilih oleh *trainee* secara lokal di komputer masing-masing. Struktur folder dan penamaan *file* tetap diatur oleh *tool* tersebut. *File* yang di-*save* tersebut nantinya akan di-*upload* oleh *tool* '*upload to server*' yang dikerjakan oleh Bapak Robert.




Gambar 3.12. *Flowchart* dari '*save new version*'

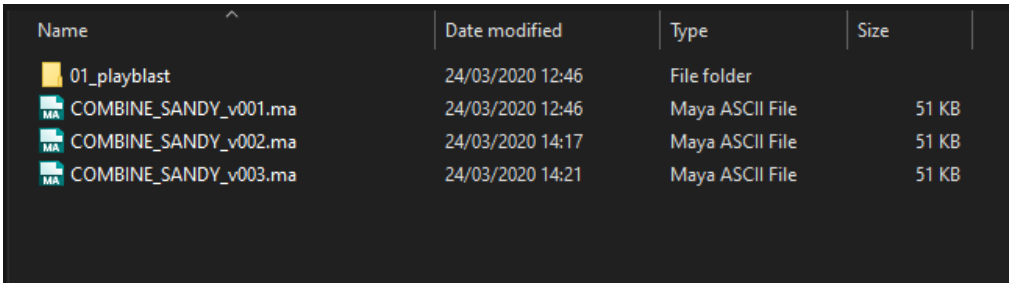
Dokumentasi Pribadi

Untuk mengimplementasikan *versioning*, penulis menggunakan *regular expressions*, atau biasa disingkat *regex*. *Regex* digunakan untuk mencari *pattern* dari *input* data. *Pattern* ini ditentukan oleh *programmer* sendiri menggunakan kombinasi dari *sequence*, *sets*, dan *metacharacters* yang tersedia sesuai dengan kebutuhan. Gambar di bawah menunjukkan pencarian pola pada nama *file* menggunakan *regex* yang berarti : Cari '_v(tiga angka).' atau '_v(tiga angka)_'.

```
if version:
    ver = 0
    for (root, dirs, files) in os.walk(self.current_dir):
        regex = re.escape(self.archive_filename) + r"(?=_)"
        ver_regex = r"((?<=_v)[0-9]{3}(?=\.))|((?<=_v)[0-9]{3}(?=_))"
        for name in files:
            if re.search(regex, name):
                try:
                    existing_ver = re.findall(ver_regex, name)[0]
```

Gambar 3.13. Potongan dari *code* yang menunjukkan penggunaan *regex*

Dokumentasi Pribadi

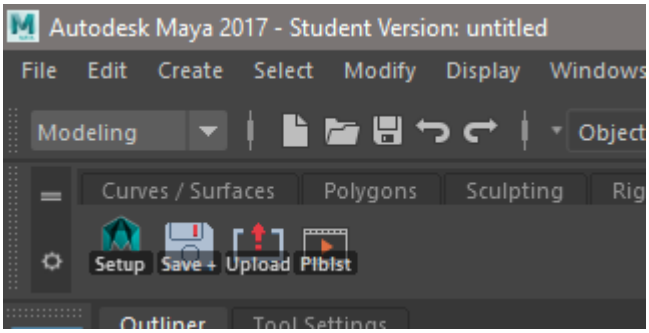



Gambar 3.14. Hasil dari penggunaan '*save new version'*

Dokumentasi Pribadi

Untuk '*quick playblast*', diperlukan *tool* 'ffmpeg' untuk konversi file AVI ke MOV. Sebelumnya *tool* ini terdapat di *server*. Artinya, sebelum didistribusikan, 'ffmpeg' juga harus di-*include* ke dalam paket instalasinya. Penulis melakukan *rerouting* tersebut diarahkan oleh Bapak Robert.

'*Upload to server*' dikerjakan oleh Bapak Robert. Integrasi ke Maya meliputi instruksi *install* juga dibuat oleh Bapak Robert. *Trainee* dapat menggunakannya hanya dengan sekali klik melalui *shelf* yang telah di-*install*.




Gambar 3.15. *Screenshot* dari *Training tools* yang telah di-*install* dalam *shelf*

Dokumentasi Pribadi

Setelah dirasa *tools* ini sudah siap untuk digunakan, semua *branch* yang dikerjakan di-*merge* ke *branch* 'master'. Bapak Robert dan Ibu Cecil (*Project Manager*) mengadakan *meeting* secara *online* antara tim *technical artist* dengan para *trainee*. Tujuannya adalah mengenalkan *tools* yang telah dikembangkan kepada penggunanya. Mulai dari instalasi hingga cara pakai. Walaupun penyampaian *tool* ini ke *trainee* sedikit terlambat, tetapi *tool* ini dapat selesai dikerjakan dan digunakan oleh para *trainee* untuk tiga minggu terakhir *training*.

**4. *Feedback***

Setelah menyelesaikan *training tools* pada minggu ke-enam magang, penulis diberi tanggung jawab untuk membantu proses *research and development*. Ditambah karena penulis bekerja secara *remote*, penulis tidak dapat melihat langsung penggunaan dari *training tools* oleh para *trainee*. Akan tetapi, penulis selalu terbuka apabila terdapat *feedback* ataupun revisi jika terdapat *bug*.

Di penghujung program *training animasi* (15 April 2020), penulis mendapat *feedback* dari para *trainee*. Mereka mengatakan bahwa *tools* tersebut berjalan lancar dan sesuai keinginan. Oleh karena penulis juga menyelesaikan magang di hari yang sama, tanggung jawab untuk pengembangan dan pemeliharaan *tools* tersebut untuk seterusnya diserahkan kepada Bapak Robert.

### 3.3.2. Kendala yang Ditemukan

Pada saat mengerjakan *render manager* dan *training tools*, penulis menemui kendala pada awal pengerjaan. Kendala tersebut adalah tidak terbiasa dengan *environment* Jasmine dan Git. Dalam pengembangan *tool* itu sendiri, penulis juga seringkali merasa pengetahuan penulis tentang Python kurang mencukupi.

Dengan kejadian pandemi Covid-19, UMN Pictures terpaksa menerapkan WFH (*Work From Home*). Bekerja secara *remote* dari rumah menjadi kendala, karena komunikasi yang lebih sulit dari biasanya dan juga tidak adanya akses secara langsung dengan *server* UMN Pictures.

### 3.3.3. Solusi Atas Kendala yang Ditemukan

Walaupun penulis tidak terbiasa dengan Jasmine pada awalnya, seiring waktu dalam pengembangan *tools*, penulis sedikit demi sedikit membaca *source code* Jasmine yang telah ditulis sebelumnya. Pada akhirnya penulis terbiasa dengan Jasmine dan *render manager* terintegrasi dengan baik di dalamnya.

Git adalah sistem yang baru bagi penulis, oleh karena itu, Bapak Robert merekomendasikan *website* interaktif https://learngitbranching.js.org, yang mengajarkan penulis dalam pengetahuan dan penggunaan Git melalui *game* dan visualisasi.

Pengetahuan tentang Python bukanlah sesuatu yang dapat dipelajari dalam satu malam. Oleh karena itu, penulis mempelajari Python sedikit demi sedikit setiap hari dengan menemui masalah dan mencoba menyelesaikannya. Penyelesaian tersebut dapat berupa eksperimen menggunakan kode baru atau penulisan ulang agar dapat menghindari terjadi masalah tersebut. Apabila penulis merasa tidak dapat menyelesaikan masalah tersebut, maka penulis selalu mendiskusikannya dengan Bapak Robert.

Untuk mengatasi kendala WFH, UMN Pictures menggunakan *plaftorm* komunikasi Microsoft Teams. Walaupun tidak se-efektif bekerja dalam kantor, dengan Teams, komunikasi yang bersifat esensial dapat tersampaikan.

Penggunaan Git juga sangat membantu dalam proses pengembangan *tools* tersebut karena semua *file* proyek disimpan secara *online*. Tidak terhubungnya penulis dengan *server* kantor membuat *tools* yang dikerjakan oleh penulis harus dapat berfungsi secara *offline*. Penulis juga banyak dibantu oleh Bapak Robert dalam menyelesaikan *training tools*.