

**NO: 01022092/INF/2021**

**RENDERING KARAKTER 3D VIRTUAL SECARA REAL-TIME MENGGUNAKAN  
METODE LIGHT ESTIMATION PADA AUGMENTED REALITY BERBASIS  
LOKASI**

**SKRIPSI**

Diajukan untuk memenuhi persyaratan penyelesaian program S-1  
Program Studi Informatika Fakultas Teknologi Industri  
Universitas Kristen Petra

Oleh :  
Kevin  
NRP: C14170003

**PROGRAM STUDI INFORMATIKA**



**FAKULTAS TEKNOLOGI INDUSTRI  
UNIVERSITAS KRISTEN PETRA  
SURABAYA  
2021**

**LEMBAR PENGESAHAN**

**SKRIPSI**

**RENDERING KARAKTER 3D VIRTUAL SECARA REAL-TIME MENGGUNAKAN  
METODE LIGHT ESTIMATION PADA AUGMENTED REALITY BERBASIS  
LOKASI**

Oleh :

Kevin

NRP: C14170003

Diterima Oleh :

Program Studi Informatika  
Fakultas Teknologi Industri  
Universitas Kristen Petra

Surabaya, 14 Juni 2021

Pembimbing I:

Pembimbing II:

(Liliana, S.T., M.Eng., Ph.D.)  
(NIP: 03-024)

(Ir. Kartika Gunadi, M.T.)  
(NIP: 88-004)

Ketua Tim Penguji:

(Stephanus A. Ananda, S.T., M.Sc. Ph.D.)  
(NIP: 93-026)

Kepala Program Studi

(Henry Novianus Palit, S.Kom., M.Kom., Ph.D.)  
(NIP: 14-001)

**LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI  
KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS**

Sebagai mahasiswa Universitas Kristen Petra, yang bertanda tangan di bawah ini,

saya :

Nama : Kevin

NRP : C14170003

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Kristen Petra Hak Bebas Royalti Non-Eksklusif (*Non-Exclusive Royalty-Free Right*) atas karya ilmiah saya yang berjudul : *Rendering Karakter 3D Virtual Secara Real-Time Menggunakan Metode Light Estimation Pada Augmented Reality Berbasis Lokasi beserta perangkat yang diperlukan*. Dengan Hak Bebas Royalti Non-Eksklusif ini Universitas Kristen Petra berhak menyimpan, mengalih-media/format-kan, mengelolanya dalam bentuk pangkalan data (*database*), mendistribusikannya, dan menampilkan/mempublikasikannya di Internet atau media lain untuk kepentingan akademis tanpa perlu meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta.

Saya bersedia untuk menanggung secara pribadi, tanpa melibatkan pihak Universitas Kristen Petra, segala bentuk tuntutan hukum yang timbul atas pelanggaran Hak Cipta dalam karya ilmiah saya ini.

Demikian pernyataan ini yang saya buat dengan sebenarnya.

Surabaya, 14 Juni 2021

(Kevin)

## KATA PENGANTAR

Segala puji dan syukur kepada Tuhan Yesus Kristus yang selalu dengan setia dan penuh kasih menyertai, membimbing, memberkati, dan memberikan kekuatan bagi peneliti hingga dapat menyelesaikan tugas akhir dengan baik. Ketika peneliti merasa tidak sanggup, tangan Tuhan yang selalu menopang dan memberikan hikmat yang luar biasa. Peneliti sangat bersyukur dan berterima kasih pada Tuhan atas setiap proses penggeraan tugas akhir ini.

Peneliti juga berterima kasih pada setiap orang yang selalu menemani, mendengarkan, memberikan semangat, dan membimbing peneliti khususnya selama mengenyam pendidikan di Universitas Kristen Petra. Terima kasih karena sudah hadir dan memberikan warna yang beragam di dalam kehidupan perkuliahan peneliti. Orang-orang yang berperan besar sehingga dapat terselesaikannya tugas akhir ini, antara lain:

1. Liliana, S.T., M.Eng., Ph.D., selaku dosen pembimbing I yang selalu memberikan waktu, tenaga, dan pikiran dalam memberikan pengarahan bagi peneliti.
2. Ir. Kartika Gunadi, M.T., selaku dosen pembimbing II yang selalu memberikan waktu, tenaga, dan pikiran dalam memberikan pengarahan bagi peneliti.
3. Segenap dosen dan staff pengajar di Program Studi Teknik Informatika dan Sistem Informasi Bisnis Universitas Kristen Petra.
4. Keluarga, Teman, dan pihak-pihak lain yang telah memberikan dukungan pada peneliti secara langsung maupun tidak langsung dalam pembuatan tugas akhir ini yang tidak dapat disebutkan satu per satu.

Peneliti menyadari penulisan skripsi ini masih belum sempurna. Oleh karena itu peneliti mengharapkan kritik dan saran dari pembaca untuk perbaikan dan penyempurnaan di penelitian selanjutnya.

Akhir kata, peneliti mohon maaf atas kekurangan yang terdapat pada penelitian ini. Peneliti berharap penelitian ini bisa memberikan tambahan wawasan bagi pembaca.

Surabaya, Juni 2021

Peneliti

## ABSTRAK

Kevin:

Skripsi

*Rendering Karakter 3D Virtual Secara Real-Time Menggunakan Metode Light Estimation Pada Augmented Reality Berbasis Lokasi*

Aplikasi *augmented reality* sudah banyak terdapat pada perangkat *mobile*, tetapi kebanyakan aplikasi *augmented reality* masih mengasumsikan bahwa sumber cahaya yang berada pada dunia virtual selalu berasal dari atas objek dan memiliki arah yang selalu ke bawah sehingga bayangan yang dihasilkan selalu tepat di bawah objek, oleh karena itu dibutuhkan metode untuk mengestimasi cahaya agar arah bayangan yang dihasilkan lebih realistik, tetapi tetap dapat dijalankan pada perangkat *mobile*. Untuk menjawab masalah di atas, digunakan metode *light estimation* pada *real-time rendering* aplikasi AR di perangkat *mobile* agar arah bayangan dari hasil *rendering* objek virtual sejajar dan searah dengan arah bayangan dari objek nyata pada lingkungannya, tetapi tetap menggunakan *resource* yang masih dapat digunakan pada perangkat *mobile*. Hasil pengujian pada penelitian ini menunjukkan bahwa arah bayangan yang dihasilkan metode estimasi cahaya pada lingkungan *indoor* cukup akurat (sekitar 33°) dan cukup ringan untuk digunakan pada perangkat *mobile*, karena perbedaan FPS dan penggunaan RAM hampir sama dengan aplikasi tanpa penggunaan metode estimasi cahaya, walaupun ada peningkatan penggunaan CPU dan baterai, tetapi cukup kecil untuk tetap dapat digunakan pada perangkat *mobile*.

Kata kunci: *real-time rendering*, estimasi cahaya, *augmented reality*, *markerless augmented reality*, aplikasi berbasis lokasi.

## **ABSTRACT**

Kevin:

Undergraduate Thesis

Real-time 3D Virtual Character Rendering Using Light Estimation on Location Based Augmented Reality

Augmented reality applications are already widely available on mobile devices, but most augmented reality applications assume that light source always comes from above the object and its direction is always downwards so that the shadow is always right under the object, therefore a method is needed to estimate light so that the direction of shadow produced is more realistic, but can still be run on mobile devices. To answer the problem, light estimation method is used in real-time rendering of AR applications on mobile devices so that the shadow direction from virtual objects rendering is parallel and in the same direction as the shadow direction of real objects in their environment, but still uses resources that can be used on mobile devices. Results in this study indicate that the direction of shadow produced by light estimation method in indoor environment is quite accurate (about 33°) and light enough to be used on mobile devices, because the difference in FPS and RAM usage is almost the same as the usage of application without the use of light estimation method, although there is an increase in CPU and battery usage, it's small enough to still work on a mobile device.

Keywords: real-time rendering, light estimation, augmented reality, markerless augmented reality, location-based application.

## DAFTAR ISI

HALAMAN JUDUL .....	i
LEMBAR PENGESAHAN.....	ii
LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI .....	iii
KATA PENGANTAR.....	iv
ABSTRAK.....	v
ABSTRACT.....	vi
DAFTAR ISI .....	vii
DAFTAR TABEL.....	ix
DAFTAR GAMBAR.....	x
DAFTAR SEGMENT.....	xiii
DAFTAR PERSAMAAN .....	xiv
DAFTAR LAMPIRAN .....	xv
1. PENDAHULUAN .....	1
1.1 Latar Belakang Masalah .....	1
1.2 Perumusan Masalah.....	4
1.3 Tujuan Skripsi .....	5
1.4 Ruang Lingkup .....	5
1.5 Metodologi Penelitian.....	7
1.6 Sistematika Penulisan.....	8
2. LANDASAN TEORI .....	9
2.1 Tinjauan Pustaka .....	9
2.2 Tinjauan Studi.....	20
3. ANALISIS DAN DESAIN SISTEM .....	23
3.1 Analisis Permasalahan.....	23
3.2 Desain Sistem .....	25
3.3 Desain Aplikasi.....	46
4. IMPLEMENTASI SISTEM.....	51
4.1 Pembuatan <i>Database</i> untuk Aplikasi Android yang digunakan .....	51
4.2 Implementasi Aplikasi Android yang digunakan .....	52
5. PENGUJIAN SISTEM .....	73

5.1 Pengujian Aplikasi .....	73
5.2 Pengujian Metode Estimasi Cahaya .....	81
5.3 Pengujian Perbedaan Penggunaan Resource.....	89
5.4 Pengujian Akurasi Pelacakan Lokasi.....	101
6. KESIMPULAN DAN SARAN .....	104
6.1 Kesimpulan .....	104
6.2 Saran.....	105
DAFTAR REFERENSI .....	106
LAMPIRAN .....	109

## DAFTAR TABEL

Tabel 4.1 Daftar Fungsi yang digunakan dalam Adegan Peta .....	52
Tabel 4.2 Daftar Fungsi yang digunakan dalam Adegan Peta .....	61
Tabel 5.1 Penggunaan CPU, RAM, dan baterai pada perangkat pertama menggunakan estimasi cahaya .....	90
Tabel 5.2 Penggunaan CPU, RAM, dan baterai pada perangkat pertama tanpa estimasi cahaya .....	91
Tabel 5.3 Rata-rata resource dan FPS pada perangkat pertama (pembulatan 2 angka di belakang koma) .....	94
Tabel 5.4 Penggunaan CPU, RAM, dan baterai pada perangkat kedua menggunakan estimasi cahaya .....	96
Tabel 5.5 Penggunaan CPU, RAM, dan baterai pada perangkat kedua tanpa estimasi cahaya .	97
Tabel 5.6 Rata-rata resource dan FPS pada perangkat kedua (pembulatan 2 angka di belakang koma) .....	100
Tabel 5.7 Akurasi lokasi yang terlacak pada perangkat pertama.....	101
Tabel 5.8 Akurasi lokasi yang terlacak pada perangkat kedua .....	102

## DAFTAR GAMBAR

Gambar 2.1 Ray tracing rendering .....	10
Gambar 2.2 Wireframe rendering.....	11
Gambar 2.3 Hidden line rendering.....	11
Gambar 2.4 Shaded rendering .....	12
Gambar 2.5 Light estimation.....	14
Gambar 2.6 Matriks translasi .....	19
Gambar 2.7 Matriks rotasi .....	19
Gambar 2.8 Matriks translasi .....	20
Gambar 3.1 Solusi masalah pencahayaan AR di perangkat mobile .....	24
Gambar 3.2 Flowchart garis besar sistem secara keseluruhan.....	26
Gambar 3.3 Use Case Diagram.....	27
Gambar 3.4 Flowchart buka daftar penanda .....	28
Gambar 3.5 Flowchart pengecekan lokasi .....	29
Gambar 3.6 Flowchart pelacakan gambar .....	30
Gambar 3. 7 Flowchart tambah penanda .....	31
Gambar 3.8 Flowchart edit penanda .....	33
Gambar 3.9 Flowchart hapus penanda .....	35
Gambar 3.10 Flowchart spawn character .....	37
Gambar 3.11 Flowchart ganti animasi .....	38
Gambar 3.12 Flowchart ganti transformasi .....	39
Gambar 3.13 Flowchart ganti model.....	41
Gambar 3.14 Flowchart ambil foto .....	43
Gambar 3.15 Flowchart reset sesi.....	44
Gambar 3.16 Flowchart buka settings .....	45
Gambar 3.17 UI adegan awal .....	46
Gambar 3.18 UI adegan peta .....	47
Gambar 3.19 UI adegan image tracking.....	48
Gambar 3.20 Logo UKP .....	48
Gambar 3.21 UI adegan AR .....	49
Gambar 3.22 Header adegan AR dan preview foto .....	49

Gambar 3.23 Model 3D karakter virtual .....	50
Gambar 4.1 Konsol Firebase .....	51
Gambar 4.2 Peraturan database .....	52
Gambar 5.1 Pengujian pada kolam gedung P, kolam jodoh, dan patung torso (cahaya dari depan objek) .....	74
Gambar 5.2 Pengujian berbagai angle pada patung torso (cahaya dari depan torso) .....	74
Gambar 5.3 Pengujian berbagai angle pada patung torso dengan karakter lain (cahaya dari depan torso) .....	75
Gambar 5.4 Pengujian berbagai angle pada kolam gedung Q (Cahaya dari depan objek).....	75
Gambar 5.5 Pengujian berbagai angle pada depan gedung Q (petraneous) (cahaya dari atas objek).....	76
Gambar 5.6 Pengujian berbagai angle pada tangga gedung Q (cahaya dari bawah tangga / luar) .....	77
Gambar 5.7 Pengujian menu utama .....	78
Gambar 5.8 Pengujian adegan peta .....	78
Gambar 5.9 Pengujian adegan image tracking .....	79
Gambar 5.10 Ganti karakter dan animasi pada adegan AR .....	80
Gambar 5.11 Import dan pengambilan foto pada adegan AR .....	80
Gambar 5.12 Perbandingan arah bayangan outdoor pada siang hari (cahaya dari kanan belakang objek) .....	82
Gambar 5.13 Perbandingan arah bayangan outdoor pada malam hari (cahaya dari belakang objek).....	83
Gambar 5.14 Perbandingan arah bayangan <i>indoor</i> pertama (cahaya dari depan kiri objek)....	84
Gambar 5.15 Perbandingan arah bayangan indoor kedua (cahaya dari belakang objek) .....	84
Gambar 5.16 Perbandingan arah bayangan indoor ketiga (cahaya dari atas objek) .....	85
Gambar 5.17 Perbandingan arah bayangan indoor keempat (cahaya dari atas objek) .....	86
Gambar 5.18 Perbandingan arah bayangan indoor kelima (cahaya dari belakang kanan objek) .....	86
Gambar 5.19 Perbandingan arah bayangan indoor keenam (cahaya dari depan kanan objek).87	87
Gambar 5.20 Perbandingan nilai realistik .....	87
Gambar 5.21 Pengujian kekurangan lain dari metode estimasi cahaya .....	88
Gambar 5.22 Pengambilan foto pada lingkungan dengan dua sumber cahaya (cahaya dari depan kanan dan kiri objek).....	89

Gambar 5.23 Penggunaan CPU, RAM, dan baterai pada perangkat pertama .....	90
Gambar 5.24 Perbandingan penggunaan CPU pada perangkat pertama.....	93
Gambar 5.25 Perbandingan penggunaan RAM dan baterai pada perangkat pertama .....	93
Gambar 5.26 Pengujian FPS pada perangkat pertama .....	94
Gambar 5.27 Perbandingan nilai rata-rata <i>resource</i> dan FPS pada perangkat pertama .....	95
Gambar 5.28 Penggunaan CPU, RAM, dan baterai pada perangkat kedua .....	95
Gambar 5.29 Perbandingan penggunaan CPU pada perangkat kedua.....	98
Gambar 5.30 Perbandingan penggunaan RAM dan baterai pada perangkat kedua .....	99
Gambar 5.31 Pengujian FPS pada perangkat kedua .....	99
Gambar 5.32 Perbandingan nilai rata-rata <i>resource</i> dan FPS pada perangkat kedua .....	100
Gambar 5.33 Lokasi pengujian .....	101
Gambar 5.34 Perbandingan akurasi pada perangkat pertama dan kedua .....	103

## DAFTAR SEGMENT

Segmen 4.1 Properti penanda.....	53
Segmen 4.2 Inisialisasi penanda.....	54
Segmen 4.3 Manajemen daftar penanda.....	55
Segmen 4.4 Cek Lokasi .....	56
Segmen 4.5 Cek gambar.....	56
Segmen 4.6 Tambah penanda.....	57
Segmen 4.7 Edit penanda.....	58
Segmen 4.8 Hapus penanda.....	59
Segmen 4.9 Manajemen kamera .....	60
Segmen 4.10 Tampilkan dan ganti posisi .....	63
Segmen 4.11 Ganti animasi.....	64
Segmen 4.12 Ganti rotasi .....	65
Segmen 4.13 Ganti skala .....	65
Segmen 4.14 Ganti model.....	66
Segmen 4.15 Import model.....	67
Segmen 4.16 Ambil foto.....	67
Segmen 4.17 Preview foto .....	69
<i>Segmen 4.18</i> reset sesi AR .....	69
Segmen 4.19 Mode debug .....	70
Segmen 4.20 Karakter debug .....	70
Segmen 4.21 Estimasi cahaya .....	71
Segmen 4.22 Shader penerima bayangan.....	71

## **DAFTAR PERSAMAAN**

Persamaan 2.1 Model waktu kontinu akselerometer.....	17
Persamaan 2.2 Model waktu kontinu giroskop .....	117
Persamaan 2.3 Model waktu kontinu magnetometer.....	117

## **DAFTAR LAMPIRAN**

Lampiran 1 Hasil kuesioner ..... 109

## 1. PENDAHULUAN

### 1.1 Latar Belakang Masalah

Pemanfaatan *smartphone* pada zaman *milenial* ini semakin banyak digunakan. Akibatnya aplikasi yang sebelumnya hanya berjalan pada *Personal Computer* (PC) harus dapat digunakan pada perangkat *mobile*, sedangkan perangkat *mobile* memiliki *resource* yang lebih terbatas dibandingkan dengan PC, oleh karena itu dibutuhkan aplikasi yang lebih ringan untuk dapat dijalankan. Salah satu metode yang dibutuhkan dalam perangkat *mobile* adalah metode untuk *rendering*.

*Rendering* dapat dimanfaatkan untuk berbagai hal, beberapa di antaranya memiliki kegunaan dalam arsitektur, *video game*, simulator, efek visual film dan TV, dan visualisasi desain, *rendering* dapat dilakukan pada dunia virtual maupun pada AR. *Rendering* pada AR dibutuhkan untuk dapat menghasilkan tekstur, pencahayaan, dan *shading* pada objek yang ditambahkan pada dunia nyata.

*Augmented Reality* (AR) adalah teknologi yang menggabungkan benda maya dua dimensi dan ataupun tiga dimensi ke dalam sebuah lingkungan nyata tiga dimensi lalu memproyeksikan benda-benda maya tersebut dalam waktu nyata. Penggunaan AR untuk mempresentasikan sesuatu pada perangkat *mobile* lebih menarik daripada menggunakan halaman web karena penggunaan AR lebih imersif dan lebih dinikmati oleh pengguna (Kowalcuk, Siepmann, & Adler, 2021). Penggunaan aplikasi AR juga dapat mengurangi beban kognitif, menghasilkan efek yang positif terhadap kesadaran konteks, keamanan (Aromaa, et al., 2020), persepsi, hubungan, dan perilaku pengguna (Nikhahemi, Knight, Nusair, & Liat, 2021). Selain itu, pemahaman pengguna juga meningkat terhadap informasi yang dipresentasikan dan aplikasi menjadi lebih menarik (Liono, Amanda, Pratiwi, & Gunawan, 2021), karena melihatkan objek yang dipresentasikan ditampilkan secara langsung kepada pengguna dapat meningkatkan motivasi pengguna (Smink, Reijmersdal, Noort, & Neijens, 2020).

AR memiliki dua jenis sistem pelacakan yang umum digunakan, yaitu pelacakan *marker-based* dan pelacakan *markerless* (Andrea, Lailiyah, Agus, & Ramadiani, 2019). *Markerless* AR memiliki banyak kelebihan dibandingkan dengan *marker-based* AR, yaitu karena penggunaannya dapat membuat aplikasi lebih dinikmati, lebih berguna, dan memudahkan pengguna untuk mendapatkan informasi yang dibutuhkan pengguna. Sehingga penggunaan *markerless* AR dapat menguntungkan baik pengguna maupun pembuat aplikasi (Qin, Peak, & Prybutok, 2021).

Salah satu strategi branding yang ditetapkan UK Petra saat ini dengan menampilkan foto lokasi *iconic* di universitas dapat dijadikan sebagai media untuk promosi universitas, yaitu dengan mengunggah foto ke sosial media seperti Instagram. Untuk meningkatkan kemenarikan foto, Teknologi *markerless AR* berbasis lokasi dapat digunakan dengan cara mengeluarkan karakter atau maskot 3D virtual untuk difoto pada lokasi-lokasinya. Penggunaan AR berbasis lokasi juga membuat penggunaan aplikasi lebih imersif (Georgiou & Kyza, 2018), lebih dinikmati, dan meningkatkan rasa ingin tahu (Harley, Lajoie, Tressel, & Jarrell, 2020).

Salah satu dari alasan terbesar dari kebanyakan pengguna yang berhenti untuk menggunakan aplikasi AR adalah karena masalah dari kelancaran penggunaan aplikasi (Alha, Koskinen, Paavilainen, & Hamari, 2019), dan kebanyakan *markerless AR* memiliki masalah akurasi (Gomez-Jauregui, Manchado, Del-Castillo-Igareda, & Otero, 2019), oleh karena itu, untuk menghasilkan aplikasi AR yang baik, sebaiknya menyelesaikan masalah-masalah yang terdapat pada aplikasi AR. Walaupun penggunaan *markerless AR* lebih menarik daripada penggunaan AR berbasis penanda, tetapi nilai kegunaannya lebih rendah (Brito & Stoyanova, 2018), salah satunya terjadi pada *markerless AR* berbasis lokasi, di mana pelacakan *indoor* memiliki akurasi yang rendah. Untuk mengatasi masalah ini, digunakan pelacakan AR cadangan yang berbasis penanda seperti pelacakan yang menggunakan penanda berupa gambar (*Image Tracking*), yang digunakan bersamaan dengan pelacakan *markerless AR* berbasis lokasi untuk meningkatkan akurasi pelacakan dalam penggunaan aplikasi AR berbasis lokasi.

Ada masalah yang terdapat pada *Rendering* aplikasi AR di perangkat *mobile*, yaitu metode *rendering* yang digunakan pada PC tidak dapat digunakan karena membutuhkan *resource* yang sangat besar, selain itu, karena harga perangkat *mobile* semakin terjangkau dan terdapat peningkatan performa dari CPU dan GPU untuk perangkat *mobile*, maka semakin banyak proses dan algoritma kompleks yang sebelumnya tidak dapat dijalankan pada perangkat *mobile* menjadi dapat dilakukan (Díaz-García, Brunet, Navazo, & Vázquez, 2018) (Roberto, Lima, Uchiyama, Teichrieb, & Taniguchi, 2019). Oleh karena itu diperlukan metode *rendering* khusus pada perangkat *mobile* yang lebih ringan agar dapat dijalankan secara *real-time*, namun hasilnya tidak berbeda jauh dari *rendering* pada PC.

Skripsi ini akan menggunakan metode light estimation untuk *real-time rendering* pada aplikasi AR di perangkat *mobile* agar memiliki hasil yang serupa dengan *rendering* pada PC, tetapi tidak menggunakan *resource* yang terlalu besar.

Ada beberapa aplikasi yang sudah dapat *merender* pada aplikasi AR di perangkat *mobile* secara *real-time*, contohnya adalah aplikasi “IKEA Place”, dan “Augment - 3D Augmented

Reality". Pada aplikasi "IKEA Place" ini, *rendering* dari objek 3D sudah memiliki tekstur dan *shading* yang baik, lalu juga sudah berjalan secara *real-time*, tetapi pencahayaan pada objek 3D masih diasumsikan sumber cahayanya berasal dari atas objek dan memiliki arah yang selalu ke bawah sehingga bayangan yang dihasilkan selalu tepat di bawah objek 3D, sehingga kurang menyatu dengan lingkungan pada dunia nyata karena pada lingkungan *outdoor* maupun *indoor*, cahaya matahari ataupun lampu tidak selalu datang dari atas objek. Pada aplikasi Augment "Augment - 3D Augmented Reality" juga sudah berjalan secara *real-time* dan sudah memiliki *shading*, tetapi objek yang *dirender* tidak menghasilkan bayangan. Kedua aplikasi di atas ini masih belum menggunakan metode untuk membuat arah bayangan yang dihasilkan oleh objek 3D virtual sejajar dan searah dengan arah bayangan dari objek nyata pada lingkungannya, sehingga bayangan yang dihasilkan masih kurang realistik. Pada skripsi ini akan dibuat metode pencahayaan yang menyesuaikan dengan cahaya dari lingkungan yang ada pada dunia nyata, sehingga bayangan yang dihasilkan sesuai dengan lingkungannya.

Ada beberapa penelitian sebelum ini yang serupa, salah satunya adalah "sistem *markerless* AR dengan estimasi sumber cahaya untuk pencahayaan langsung" (Frahm, Koeser, Grest, & Koch, 2005). Penelitian ini sudah dapat menghasilkan bayangan yang realistik pada *markerless* AR, tetapi metode ini sangat tidak sesuai untuk digunakan pada perangkat *mobile*, karena membutuhkan lebih dari satu kamera, membutuhkan kamera *fish-eye*, dan membutuhkan waktu setelah kamera dipindahkan. Penelitian lainnya ada yang menggunakan Deep Learning untuk mengestimasi cahaya (Marques, Clua, & Vasconcelos, 2018) (Kán & Kafumann, 2019). Pada penelitian ini, metode yang digunakan tidak memerlukan peralatan khusus atau pengetahuan sebelumnya tentang lingkungannya, dan dapat digunakan pada lingkungan *indoor* maupun *outdoor*, tetapi metode ini memerlukan data set dan pelatihan terlebih dahulu sebelum dapat digunakan.

Penelitian serupa pada topik *rendering* karakter 3D virtual adalah "pembuatan interaksi animasi karakter 3D untuk AR" (Pantuwong, 2016). Penelitian ini menghasilkan *rendering* pada perangkat dan menggunakan penanda pada AR untuk berinteraksi dengan animasi karakter 3D, tetapi animasi yang dapat digunakan terbatas pada penanda, tidak ada parameter yang dapat diatur pada animasi, harus menggunakan penanda khusus, dan tidak ada metode untuk menyamakan pencahayaan dengan dunia nyata. Metode *rendering* pada penelitian lainnya adalah metode *cloud-to-end* (Zhang, Zhang, Yin, Zhou, & Pan, 2020). Metode ini dapat menjalankan *rendering* pada server di *cloud*, sehingga hasil *rendering* yang didapatkan oleh pengguna berkualitas lebih tinggi daripada hasil *rendering* yang dapat diproses oleh perangkat

pengguna. Hasil *rendering* juga dapat dibagikan ke berbagai pengguna, sehingga berguna untuk digunakan pada aplikasi yang akan digunakan oleh banyak pengguna. Tetapi metode ini membutuhkan koneksi internet yang cukup cepat dan stabil agar dapat dijalankan secara *real-time*. Kelebihan dari metode *cloud-to-end* juga tidak terlalu dibutuhkan dalam penelitian ini.

Dari segi AR berbasis lokasi, penelitian yang serupa adalah “penggunaan *cloud* pada AR untuk pencarian lokasi” (Meenakshi, Vasudevan, Ritesh, & Santhosh, 2015). Penelitian ini menggabungkan teknologi *cloud* pada AR untuk pembuatan aplikasi pencarian lokasi berbasis web pada perangkat *mobile*, tetapi aplikasi tidak dapat menambah lokasi objek yang ditampilkan, dan dibutuhkan server yang selalu menyala untuk menyediakan informasi kepada aplikasi.

Oleh karena itu pada skripsi ini akan dilakukan penelitian untuk memodifikasi *shader* dan menggunakan *library* ARCore untuk menyesuaikan pencahayaan *rendering* karakter 3D virtual yang memiliki animasi pada perangkat *mobile* android agar arah bayangan yang dihasilkan oleh *rendering* karakter 3D virtual sejajar dan searah dengan arah bayangan dari objek nyata pada lingkungannya dengan menggunakan metode yang cukup ringan dan dapat digunakan oleh kebanyakan *smartphone*, yaitu dengan metode *light estimation*. Setelah itu menggabungkannya dengan aplikasi AR berbasis lokasi, sehingga menghilangkan kebutuhan untuk menggunakan penanda khusus sekaligus menambahkan kemampuan untuk menambah animasi yang dapat digunakan. Objek dan lokasi yang digunakan untuk *rendering* juga dapat ditambahkan langsung dan disimpan pada perangkat, sehingga tidak membutuhkan server khusus untuk menyimpan data objek dan lokasi.

## 1.2 Perumusan Masalah

Berdasarkan penjelasan yang terdapat di latar belakang, maka dapat dirumuskan masalah dalam skripsi ini sebagai berikut:

- Apakah memodifikasi *shader* dan menggunakan metode *light estimation* di *library* ARCore dapat membuat arah bayangan yang dihasilkan oleh *rendering* karakter 3D virtual sejajar dan searah dengan arah bayangan dari objek nyata pada lingkungannya ?
- Seberapa besar perbedaan penggunaan *resource* pada *rendering* yang menggunakan dan tidak menggunakan *light estimation* pada beberapa jenis perangkat *mobile* ?
- Seberapa akurat pelacakan menggunakan sensor lokasi pada beberapa jenis perangkat *mobile* ?

### **1.3 Tujuan Skripsi**

Tujuan dari penulisan skripsi ini adalah untuk mengganti metode pencahayaan dari *rendering* objek 3D pada *Augmented Reality* menggunakan *light estimation* pada ARCore agar arah bayangan yang dihasilkan oleh *rendering* karakter 3D virtual sejajar dan searah dengan arah bayangan dari objek nyata pada lingkungannya, sehingga lebih realistik, dan menggabungkannya dengan *markerless AR* berbasis lokasi dengan menggunakan sensor lokasi dan *image tracking* agar dapat digunakan sebagai metode pengenalan dan pelacakan lokasi yang digunakan untuk menampilkan karakter 3D virtual.

### **1.4 Ruang Lingkup**

Ruang lingkup pada skripsi ini dibatasi pada:

- Aplikasi ini menggunakan teknologi *Markerless Augmented Reality* yang pencahayaannya menggunakan metode *light estimation*, yaitu metode untuk mengestimasi cahaya pada *library* ARCore, yang digunakan bersamaan dengan *shader* yang dimodifikasi dan dikerjakan pada skripsi ini agar arah bayangan yang dihasilkan oleh *rendering* karakter 3D virtual sejajar dan searah dengan arah bayangan dari objek nyata pada lingkungannya. Untuk menampilkan objek 3D maupun melacak posisi dan gerakan digunakan metode *Simultaneous localization and mapping* (SLAM) pada *library* ARCore, dan untuk pelacakan rotasi digunakan sensor *Inertial Measuring Unit* (IMU).
- *Rendering* karakter 3D virtual dilakukan pada *Augmented Reality* berbasis lokasi untuk *mobile devices* Android, yang menggunakan *library* Mapbox Maps SDK untuk melacak lokasi pengguna berdasarkan sensor lokasi, dan menggunakan pelacakan gambar pada *library* ARCore untuk pelacakan gambar 2D sebagai cadangan pelacakan lokasi.
- *Output* akhir yang dihasilkan dari aplikasi berupa *rendering* karakter 3D virtual yang memiliki bayangan sejajar dan searah dengan arah bayangan dari objek nyata pada lingkungannya, yang ditampilkan pada lokasi yang sudah ditentukan, melalui layar pada perangkat. Terdapat tujuh karakter 3D model yang disediakan pada aplikasi. Pengguna juga dapat memasukkan data 3D modelnya sendiri (berupa *file* dengan ekstensi OBJ) untuk *dirender* pada aplikasi.
- Aplikasi dibuat menjadi dua versi, yaitu untuk admin dan untuk pengguna. Admin dapat menentukan lokasi, nama, deskripsi, dan karakter *default* di mana

karakter dikeluarkan, dan dapat melakukan apa pun yang dapat dilakukan pengguna. Karakter yang dimaksud adalah karakter 3D virtual yang dapat digunakan untuk foto bersama (karakter dan orang) pada lokasi yang sudah ditentukan.

- Pengguna dapat mengambil foto bersama dengan karakter 3D virtual yang pencahayaannya *dirender* secara *real-time* sehingga karakter memiliki bayangan yang sesuai dengan lingkungannya. Pengguna juga dapat mengganti animasi dari karakter dengan menekan tombol untuk ganti animasi pada *user interface* aplikasi.
- Pengambilan foto dilakukan dengan menuju lokasi yang sudah ditentukan oleh admin, lalu menekan penanda pada peta untuk berpindah ke mode AR. Pada mode AR, pengguna dapat mengeluarkan dan mengatur karakter 3D virtual sesuai dengan keinginan, dan dapat mengambil foto dengan cara mengarahkan kamera perangkat ke karakter dan menekan tombol untuk *capture* pada *user interface* aplikasi.
- Jika lokasi yang ditentukan berada di dalam gedung (indoors) dan terdapat lebih dari satu lantai, maka admin dapat menyediakan gambar yang dapat digunakan untuk pelacakan cadangan yaitu *image tracking* sebagai cara untuk berpindah ke mode AR.
- Pada kedua versi aplikasi ini juga dapat mengatur transformasi (translasi, rotasi, dilatasi) dari karakter, melihat daftar lokasi yang sudah ditentukan oleh admin, dan melihat propertinya berupa lokasi, nama, deskripsi, dan karakter *default*.
- Pengujian dilakukan pada perangkat dengan Chipset Exynos 7885 Octa-core (2x2.2 GHz & 6x1.6 GHz), GPU Mali-G71, RAM 4GB, dan perangkat dengan Chipset Mediatek Helio G90T Octa-core (2x2.05 GHz & 6x2.0 GHz), GPU Mali-G76 MC4, RAM 6GB.
- Pengujian penggunaan metode *light estimation* terhadap realistik atau tidaknya bayangan *rendering* objek 3D pada *Augmented Reality* dilakukan dengan cara melihat apakah hasil bayangan objek 3D yang *dirender* dengan metode *light estimation* pada aplikasi sejajar dan searah dengan bayangan dari objek nyata pada lingkungannya, dan membandingkannya dengan hasil bayangan objek 3D yang *dirender* tanpa metode *light estimation*.

- Pengujian seberapa besar perbedaan penggunaan resource pada rendering objek 3D dilakukan dengan cara membandingkan penggunaan CPU, penggunaan RAM, waktu yang dibutuhkan untuk *merender* (*Frames per Second*), dan penggunaan baterai pada *rendering* yang menggunakan dan tidak menggunakan *light estimation*.
- Pengujian seberapa akuratnya pelacakan lokasi dilakukan dengan cara mengukur rata-rata jarak antara posisi penanda yang dikeluarkan di lokasi pengguna berdasarkan sensor lokasi dengan lokasi di mana seharusnya penanda dikeluarkan, pada setiap perangkat yang digunakan.
- Perangkat yang digunakan harus dapat menggunakan ARCore, yaitu Android dengan versi 7.0 atau terdapat pada daftar perangkat yang berada pada halaman web <https://developers.google.com/ar/discover/supported-devices>, memiliki sensor lokasi dan IMU.
- Aplikasi dibuat dengan menggunakan *Unity Engine* dengan *script* yang menggunakan bahasa pemrograman C#, menggunakan *AR Foundation Framework* sebagai penghubung ke platform ARCore, dan menggunakan Mapbox Maps SDK untuk Unity.

## 1.5 Metodologi Penelitian

Langkah-langkah yang akan dilakukan dalam penggerjaan penelitian ini adalah sebagai berikut:

1. Studi literatur tentang:
  - Markerless Augmented Reality
  - ARCore dan Light Estimation
  - Mapbox Maps SDK untuk Unity
  - AR Foundation pada Unity
2. Perencanaan dan Pembuatan Perangkat Lunak:
  - Pemilihan dan pengambilan data *3D Model*
  - Perencanaan dan pembuatan *User Interface*
  - Pembuatan karakter 3D virtual dan implementasi AR
3. Pengujian dan Analisis Perangkat Lunak
  - Pengujian program yang telah dibuat
  - Analisis hasil output dari program

#### 4. Pembuatan Laporan

##### **1.6 Sistematika Penulisan**

Sistematika penulisan dalam laporan skripsi ini terdiri dari beberapa bab, yaitu:

BAB 1 : PENDAHULUAN

Bab ini berisi penjelasan yang berkaitan dengan latar belakang masalah, perumusan masalah, tujuan skripsi, ruang lingkup, metodologi penelitian dan sistematika penulisan yang digunakan.

BAB 2 : LANDASAN TEORI

Bab ini berisi penjelasan yang berkaitan dengan teori dan metode yang digunakan dalam penelitian atau penggerjaan skripsi.

BAB 3 : ANALISIS DAN DESAIN SISTEM

Bab ini berisi penjelasan yang berkaitan dengan gambaran besar dari penelitian yang akan dilakukan melalui analisis dan perancangan desain sistem secara keseluruhan..

BAB 4 : IMPLEMENTASI SITEM

Bab ini berisi penjelasan yang berkaitan dengan implementasi aplikasi atau sistem berdasarkan analisa dan perancangan desain sistem yang sudah dibuat pada Bab 3.

BAB 5 : PENGUJIAN SISTEM

Bab ini berisi penjelasan yang berkaitan dengan hasil pengujian dari aplikasi atau sistem yang sudah diimplementasikan atau dibuat pada Bab 4.

BAB 6 : KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan yang didapat dari hasil pengujian yang sudah dilakukan dan beberapa saran untuk pengembangan penelitian selanjutnya.

## 2. LANDASAN TEORI

### 2.1 Tinjauan Pustaka

#### 2.1.1 Markerless Augmented Reality

*Markerless Augmented Reality* adalah jenis *Augmented Reality* yang tidak memerlukan pengetahuan awal apa saja tentang lingkungan penggunanya untuk menampilkan dan meletakan konten 3D virtual pada sebuah tempat dan menahannya pada titik tetap di dalam ruang (Schechter, 2020).

*Augmented Reality* (AR) adalah salah satu teknologi *Extended Reality* yang dapat menampilkan pandangan *real-time* secara langsung atau tidak langsung dari sebuah lingkungan fisik dunia nyata yang secara mulus telah digabungkan dengan menambah objek atau informasi digital (suara, video, grafik, teks, atau lokasi geografis) yang bersifat virtual dan mempresentasikan hasil penggabungannya kepada pengguna (Jinyu, et al., 2019) (Gomez-Jauregui, Manchado, Del-Castillo-Igareda, & Otero, 2019).

*Extended Reality* (XR) adalah istilah untuk semua lingkungan yang menggabungkan lingkungan nyata dan virtual, dan interaksi antara manusia dan mesin yang dibuat oleh teknologi komputer dan alat yang dapat dipakai (Greenwold, 2003).

Manusia secara tidak sadar menangkap tanda-tanda tentang bagaimana benda atau makhluk hidup dicahayai di lingkungannya. Saat objek virtual kehilangan bayangan, atau memiliki bahan mengkilap yang tidak mencerminkan ruang sekitarnya, pengguna dapat merasakan objek tersebut tidak cukup cocok dengan pemandangan tertentu meskipun mereka tidak dapat menjelaskan alasannya. Inilah mengapa *merender* objek AR agar sesuai dengan pencahayaan dalam sebuah pemandangan sangat penting untuk pengalaman yang imersif dan lebih realistik (Google LLC, 2020).

#### 2.1.2 Rendering

*Rendering* adalah proses dari membangun gambar dari sebuah model (2D atau 3D), ataupun dari sebuah berkas adegan, melalui program komputer. Sebuah berkas adegan pada *rendering* terdiri dari objek-objek dalam sebuah struktur data, yang dapat berupa objek geometri, sudut pandang, tekstur, pencahayaan, dan informasi bayangan sebagai sebuah deskripsi dari adegan virtual. Data yang terisi dalam berkas adegan kemudian melewati program *rendering* untuk diproses dan menjadi hasil keluaran untuk sebuah gambar digital (Akenine-Möller, Haines, & Hoffman, 2018).

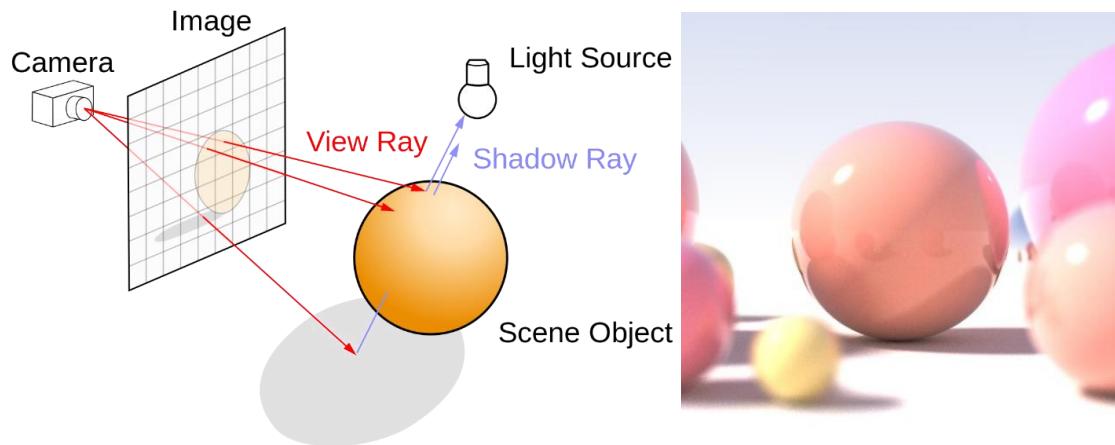
*Rendering* memiliki banyak pemanfaatan dan aplikasi yang dapat digunakan pada banyak bidang, seperti pada pemrograman game, arsitektur, simulator, perfilman, efek spesial pada tayangan televisi, dan visualisasi desain. *Rendering* pada bidang-bidang tersebut memiliki perbedaan, terutama pada fitur dan metode *renderingnya* (Listianawati, Ayu, Arista, Stephanie, & Dyastuti, 2011).

Walaupun detail-detail teknis dalam metode *rendering* bervariasi, proses umum dalam memproduksi sebuah gambar dua dimensi dari gambar tiga dimensi yang disimpan dalam sebuah berkas adegan, ditangani oleh sebuah peralatan *rendering*, seperti GPU. GPU adalah peralatan yang dibangun dengan tujuan khusus untuk mempermudah CPU dalam menangani kalkulasi *rendering* yang kompleks.

#### 2.1.2.1 Metode Rendering

Saat ini, ada beberapa metode *rendering* yang dapat digunakan, yaitu :

- *Ray tracing rendering*



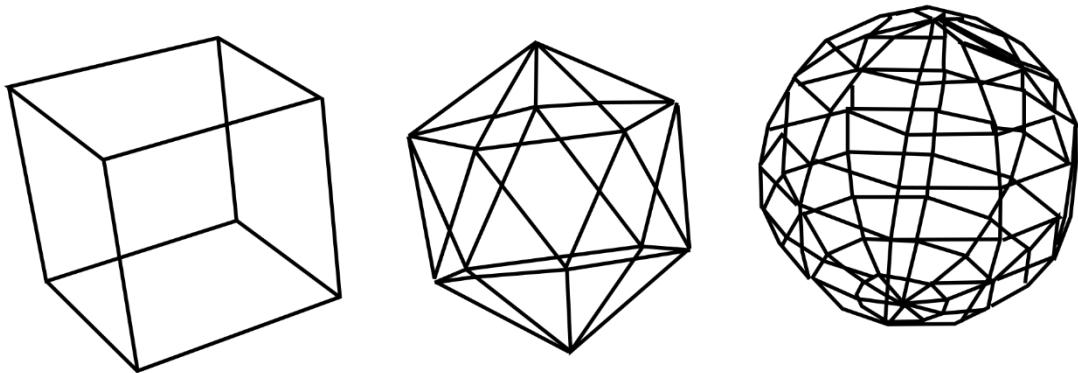
Gambar 2.1 *Ray tracing rendering*

Sumber : *Ray tracing (graphics)*. (n.d.). [https://en.wikipedia.org/wiki/Ray\\_tracing\\_\(graphics\)](https://en.wikipedia.org/wiki/Ray_tracing_(graphics))

*Ray tracing* sebagai sebuah metode *rendering* pertama kali digunakan pada tahun 1980 untuk pembuatan gambar tiga dimensi. Ide dari metode *rendering* ini sendiri berasal dari percobaan Rene Descartes, di mana ia menunjukkan pembentukan pelangi dengan menggunakan bola kaca berisi air dan kemudian meruntut kembali arah datangnya cahaya dengan memanfaatkan teori pemantulan dan pembiasan cahaya yang telah ada saat itu. Metode *rendering* ini diyakini sebagai salah satu metode yang menghasilkan gambar bersifat paling *photorealistic*. Konsep dasar dari metode ini adalah meruntut proses yang dialami oleh sebuah cahaya dalam perjalanannya dari sumber cahaya hingga layar dan memperkirakan warna

macam apa yang ditampilkan pada *pixel* tempat jatuhnya cahaya. Proses tersebut akan diulang hingga seluruh *pixel* yang dibutuhkan terbentuk (Listianawati, Ayu, Arista, Stephanie, & Dyastuti, 2011). Kelemahan terbesar dari metode *ray tracing* adalah performanya. Hingga akhir 2010-an, *ray tracing* secara *real-time* biasanya dianggap mustahil pada perangkat keras konsumen untuk tugas-tugas non trivial (Listianawati, Ayu, Arista, Stephanie, & Dyastuti, 2011).

- *Wireframe rendering*

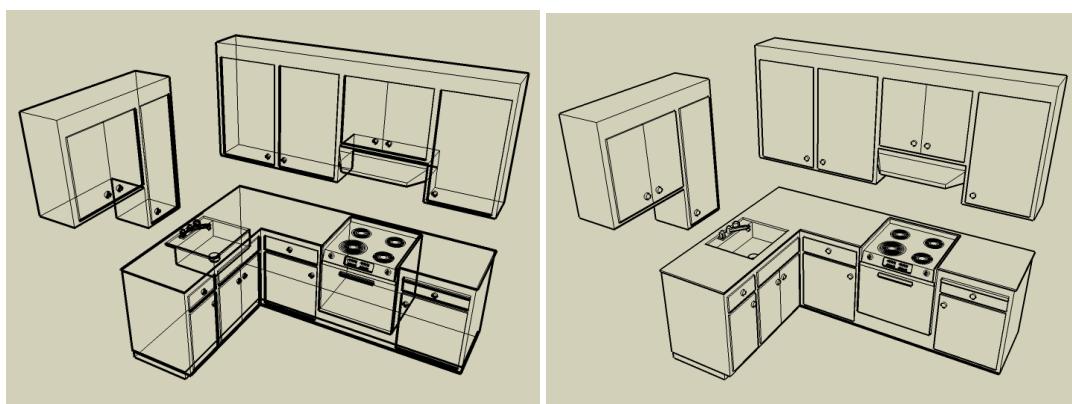


Gambar 2.2 *Wireframe rendering*

Sumber : *Wire-frame model*. (n.d.). [https://en.wikipedia.org/wiki/Wire-frame\\_model](https://en.wikipedia.org/wiki/Wire-frame_model)

*Wireframe* yaitu Objek 3D dideskripsikan sebagai objek tanpa permukaan. Pada *wireframe rendering*, sebuah objek dibentuk hanya terlihat garis-garis yang menggambarkan *edges* dari sebuah objek. Metode ini dapat dilakukan oleh sebuah komputer dengan sangat cepat, hanya kelebihannya adalah tidak adanya permukaan, sehingga sebuah objek terlihat transparan. Sehingga sering terjadi kesalahpahaman antara sisi depan dan sisi belakang dari sebuah objek (Listianawati, Ayu, Arista, Stephanie, & Dyastuti, 2011).

- *Hidden line rendering*



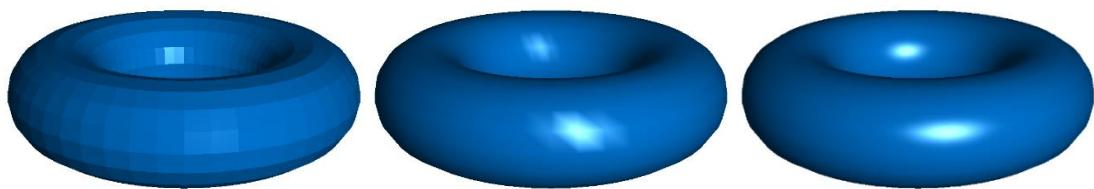
Gambar 2.3 *Hidden line rendering*

Sumber : *Hidden Line*. (n.d.).

<https://sites.google.com/site/sagesuwiki/terminology/glossary/h/hidden-line>

Metode ini menggunakan fakta bahwa dalam sebuah objek, terdapat permukaan yang tidak terlihat atau permukaan yang tertutup oleh permukaan lainnya. Dengan metode ini, sebuah objek masih direpresentasikan dengan garis-garis yang mewakili sisi dari objek, tapi beberapa garis tidak terlihat karena adanya permukaan yang menghalanginya. Metode ini lebih lambat dari *wireframe rendering*, tapi masih dikatakan relatif cepat. Kelemahan metode ini adalah tidak terlihatnya karakteristik permukaan dari objek tersebut, seperti warna, kilauan (*shininess*), tekstur, pencahayaan, dll (Listianawati, Ayu, Arista, Stephanie, & Dyastuti, 2011).

- *Shaded rendering*



Gambar 2.4 *Shaded rendering*

Sumber : *Shading*. (n.d.). <https://en.wikipedia.org/wiki/Shading>

Pada metode ini, komputer diharuskan untuk melakukan berbagai perhitungan baik pencahayaan, karakteristik permukaan, *shadow casting*, dll. Metode ini menghasilkan citra yang sangat realistik, tetapi kelemahannya adalah lama waktu *rendering* yang dibutuhkan (Listianawati, Ayu, Arista, Stephanie, & Dyastuti, 2011).

#### 2.1.2.2 Light Estimation

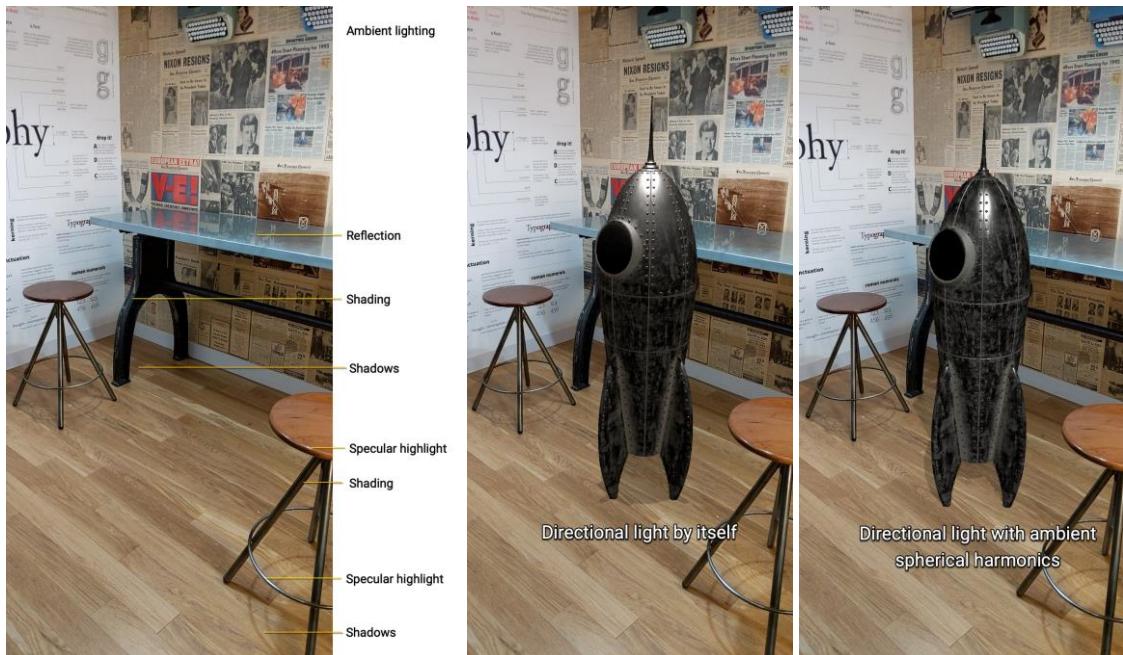
Dalam kasus grafik 3D, adegan dapat dibuat sebelumnya atau dibuat dalam waktu nyata. Untuk menghasilkan adegan yang paling realistik, adegan dibuat sebelumnya menggunakan *pre-rendering*, yaitu proses yang lambat dan intensif secara komputasi, yang biasanya digunakan untuk pembuatan film, di mana adegan dapat dibuat sebelum dibutuhkan. Video game 3D dan aplikasi lain yang harus bisa membuat adegan secara dinamis harus menggunakan *Real-time rendering*, yaitu proses membuat gambar dengan cepat dalam waktu nyata pada komputer (Akenine-Möller, Haines, & Hoffman, 2018).

Aplikasi AR merupakan adegan yang dinamis, sehingga membutuhkan metode *real-time rendering*, tetapi karena metode ini dijalankan dalam waktu nyata, maka adegan yang dihasilkan biasanya kurang realistik, sedangkan di dalam aplikasi AR biasanya diharapkan / diinginkan untuk menghasilkan *rendering* yang serealistik mungkin, semakin realistik hasil *rendering* yang

didapatkan maka semakin baik. Oleh karena itu ada beberapa cara yang dapat dilakukan dengan tujuan untuk membuat hasil dari *real-time rendering* menjadi lebih realistik, salah satu caranya adalah dengan mengestimasi sumber cahaya pada suatu adegan.

*Light Estimation* adalah metode untuk mengestimasi sumber cahaya pada suatu adegan yang dapat mewakili berbagai sumber cahaya, dengan intensitas dan arah yang berbeda (Marques, Clua, & Vasconcelos, 2018). Salah satu cara untuk menggunakan metode *light estimation* adalah dengan menggunakan *lighting estimation API* pada dalam *library ARCore*. API (*Application Programming Interface*) adalah penerjemah komunikasi antara klien dengan server untuk menyederhanakan implementasi dan perbaikan perangkat lunak. Bisa diartikan juga sebagai sekumpulan perintah, fungsi, serta protokol yang dapat digunakan oleh *programmer* saat membangun perangkat lunak untuk sistem operasi tertentu. API memungkinkan *programmer* untuk menggunakan fungsi standar untuk berinteraksi dengan sistem operasi (Fisher, 1989).

*Input* yang diterima oleh *lighting estimation API* adalah gambar berupa foto, yang selanjutnya digunakan dan dianalisis untuk mencari cahaya ambient, bayangan, shading, specular highlights, dan refleksi, yang kemudian digunakan untuk menghasilkan *output*. *Output* dari API berupa informasi mendetail tentang estimasi pencahayaan, yaitu intensitas cahaya *ambient*, warna cahaya *ambient*, arah cahaya utama, intensitas cahaya utama, dan warna cahaya utama, dalam suatu adegan tertentu (Google LLC, 2020). Estimasi arah dari cahaya didapatkan melalui *image processing* pada API, lalu, Intensitas dan warna dari cahaya didapatkan dengan cara memperhitungkan rata-rata dari kecerahan dan warna keseluruhan gambar yang diproses (Jakl, 2017).



Gambar 2.5 *Light estimation*

Sumber : *Using ARCore to light models in a scene.* (n.d.).

<https://developers.google.com/ar/develop/unity/light-estimation>

Informasi yang didapatkan dari *output lighting estimation API* ini kemudian digunakan pada saat *merender* objek virtual untuk menerangi objek dalam kondisi yang sama dengan pemandangan tempat objek tersebut ditempatkan, sehingga dapat menghasilkan cahaya dengan arah dan intensitas yang sama, dan hal ini membuat objek tersebut terasa lebih realistik dan meningkatkan pengalaman imersif bagi pengguna (Google LLC, 2020).

#### 2.1.2.3 *Shader*

Penggunaan informasi yang didapatkan dari metode *light estimation* ini dilakukan dengan cara mengatur pencahayaan pada dunia virtual, yang digunakan untuk menerangi objek virtual pada suatu adegan, dapat mewakili pencahayaan pada dunia nyata. Data pencahayaan yang telah diatur ini kemudian digunakan pada *shader* yang telah dimodifikasi pada penelitian ini.

*Shader* adalah program kecil yang dijalankan pada GPU, yang digunakan untuk membuat bayangan, mengubah warna dalam adegan 3D, dan melakukan berbagai fungsi khusus di berbagai bidang dalam kategori grafik komputer lainnya, sesuai dengan algoritma yang dibuat. Dalam pengertian dasar, *shader* hanyalah program yang mengubah *input* menjadi *output*. *Shader* juga merupakan program yang sangat terisolasi karena tidak diizinkan untuk berkomunikasi antara satu sama lain, satu-satunya komunikasi yang mereka miliki adalah

melalui *input* dan *output* (Vries, n.d.). Pada data pencahayaan global, terdapat variabel arah, intensitas, dan warna cahaya yang dapat digunakan pada *shader* untuk memperhitungkan warna pada objek 3D virtual yang akan *dirender*.

Dalam *shader* yang telah dimodifikasi pada penelitian ini, *input* yang digunakan adalah sumber cahaya berupa satu *directional light*. *Directional light* adalah cahaya yang hanya menuju ke satu arah, cahaya ini dibuat seperti datang dari tempat yang sangat jauh sehingga semua cahaya yang dihasilkannya paralel. Cahaya ini kemudian digunakan untuk menghasilkan output berupa satu bayangan *hard shadows* pada bidang transparan di bawah objek 3D virtual. *Hard shadows* adalah bayangan dengan sisi yang tajam, dan hanya menggunakan status biner saja untuk mengecek sebuah titik tertentu berada pada area bayangan atau tidak (Suni & Fathoni, 2016), sehingga proses untuk menghasilkan bayangan cukup ringan untuk dijalankan pada perangkat *mobile*.

Satu *directional light* digunakan sebagai sumber cahaya karena dapat mewakili seluruh pencahayaan pada kebanyakan adegan (kebanyakan objek hanya memiliki satu bayangan) dan dapat menghasilkan bayangan yang cukup ringan untuk digunakan pada perangkat *mobile* (karena *shader* hanya memiliki satu *input*).

### 2.1.3 Metode Pelacakan

Pelacakan adalah proses yang penting dalam teknologi Augmented Reality (AR) yang dibutuhkan untuk dijalankan secara *real-time* untuk menyesuaikan objek nyata dan virtual dengan tepat, sehingga kedua objek tersebut tampak berada di dunia yang sama. Pelacakan tanpa penanda telah dikembangkan untuk mengatasi keterbatasan pada pelacakan berbasis penanda konvensional di AR. Dengan menangkap lingkungan nyata untuk menghasilkan ciri, pelacakan tanpa penanda akan mengenali ciri ini untuk melapisi objek virtual di atas ciri yang diambil. Ciri-ciri tersebut telah dilacak secara *real-time* oleh perangkat tampilan, berdasarkan lingkungan nyata. (Anggara, et al., 2020). Metode pelacakan tanpa penanda yang digunakan pada aplikasi AR adalah metode *Simultaneous Localization And Mapping*.

Pelacakan lokasi yang digunakan untuk aplikasi AR berbasis lokasi adalah Geo AR atau *Gravimetric AR*, yaitu metode yang menggunakan posisi perangkat untuk melacak lokasinya. Teknik ini tidak menggunakan *image processing*, melainkan menggunakan GPS, *Wifi*, *Bluetooth*, atau berbagai macam sensor lokasi lainnya untuk menentukan posisi perangkat (Michel, Genevès, Fourati, & Layaïda, 2018).

### **2.1.3.1 Simultaneous Localization And Mapping (SLAM)**

SLAM adalah proses yang secara *real-time* dapat membangun peta dari sebuah lingkungan yang tidak diketahui dan pada saat yang bersamaan menggunakan peta tersebut untuk menentukan lokasinya (Moezzi, Krcmarik, Hlava, & Cýrus, 2020). Dalam AR, aplikasi memerlukan inisialisasi yang cepat dan dengan skala yang akurat. Menurut penggunaan sensor yang berbeda, teknik SLAM dapat dibagi menjadi VSLAM (Visual SLAM), VISLAM (Visual-Inertial SLAM), dan sebagainya (Jinyu, et al., 2019).

VISLAM adalah teknologi yang menggunakan sensor visual dan inersia (sensor *Inertial Measuring Unit*) untuk menyimpulkan pose perangkat dan peta pemandangan di lingkungan yang tidak diketahui. Sebaliknya, VSLAM hanya menggunakan sensor visual untuk memperkirakan pose kamera dan struktur pemandangan. Informasi inersia dimodelkan oleh navigasi inersia dan dapat menggantikan cacat informasi visual. Jadi dengan menggabungkan informasi visual dan inersia, sistem VISLAM umumnya bisa lebih baik daripada sistem VSLAM dalam situasi yang sama (Jinyu, et al., 2019).

### **2.1.3.2 Inertial Measuring Unit (IMU)**

*Inertial Measuring Unit* adalah sensor elektronik yang tersusun dari tiga sensor *Micro-electro-mechanical systems*, yaitu akselerometer 3-axis, giroskop 3-axis dan magnetometer 3-axis. IMU bertujuan untuk mengukur kecepatan, orientasi, dan gaya gravitasi dari perangkat (Michel, Genevès, Fourati, & Layaïda, 2018). Perangkat *mobile* seperti *smartphone* pada umumnya memiliki sensor IMU untuk melacak rotasi dari perangkat, sehingga ideal untuk digunakan dengan SLAM pada aplikasi AR (Jinyu, et al., 2019).

*Micro-electro-mechanical systems (MEMS)* adalah struktur peralatan elektro-mekanik terdiri dari sensor mikro, aktuator mikro dan peraga pendukung lainnya di dalam ukuran miniatur seukuran rangkaian *Integrated Circuit (IC)*. *Intergrated Circuit* adalah komponen dasar yang terdiri dari resistor, transistor dan lain-lain.

Untuk mendapatkan informasi inersial dari sensor sensor-sensor MEMS yang terdapat pada sensor IMU, dibutuhkan untuk menghitungnya dalam model waktu kontinu, karena berbagai fenomena alam (misalnya gerakan benda, aliran arus listrik) berlangsung dengan lancar dari waktu ke waktu. Untuk menghitung informasi inersial dari model waktu kontinu, dibutuhkan untuk mengetahui bias dan *noise* dari sensor-sensor tersebut. *Bias instability (drift atau offset)* merupakan salah satu jenis deviasi (*error*) pengukuran, yang merupakan fluktuasi nilai pengukuran kecepatan sudut yang terjadi pada temperatur konstan ketika sensor giroskop

dalam keadaan tidak berotasi atau berotasi dengan kecepatan konstan yang diketahui (Mayditia & Prabowo, 2008). *Sensor noise* adalah *output* dari sensor yang tidak berhubungan dengan *input* sensor. Idealnya, satu-satunya *noise* yang berasal dari sensor adalah *thermal noise* yang timbul dari gerakan di dalam sensor (Masi, 2020). Berikut ini adalah penjelasan dari masing-masing sensor :

- **Akselerometer**

Akselerometer 3-axis mengukur akselerasi perangkat, termasuk gravitasi dan akselerasi eksternal dalam  $m.s^{-2}$ :  $acc = [acc_x \ acc_y \ acc_z]^T$ . Model waktu kontinu untuk akselerometer dapat ditulis:

$$acc = acc_r + acc_b + acc_n, \quad (2.1)$$

Di mana:

$acc$  adalah jumlah gravitasi dan percepatan eksternal benda yang diukur dengan akselerometer.

$acc_r$  adalah jumlah gravitasi dan percepatan eksternal benda yang sebenarnya.

$acc_b$  adalah bias akselerometer.

$acc_n$  adalah *noise* akselerometer.

(Michel, Genevès, Fourati, & Layaïda, 2018)

- **Giroskop**

Giroskop 3-axis mengukur kecepatan sudut perangkat dalam  $rad.s^{-1}$ :  $gyr = [gyr_x \ gyr_y \ gyr_z]^T$ . Model waktu kontinu untuk giroskop dapat ditulis:

$$gyr = gyr_r + gyr_b + gyr_n, \quad (2.2)$$

Di mana:

$gyr$  adalah kecepatan sudut yang diukur dengan giroskop.

$gyr_r$  adalah kecepatan sudut yang sebenarnya.

$gyr_b$  adalah bias giroskop.

$gyr_n$  adalah *noise* giroskop.

(Michel, Genevès, Fourati, & Layaïda, 2018)

- **Magnetometer**

Magnetometer 3-axis mengukur medan magnet pada perangkat dalam *micro-tesla* ( $\mu T$ ):  $mag = [mag_x \ mag_y \ mag_z]^T$ . Model waktu kontinu untuk magnetometer dapat ditulis:

$$mag = mag_r + mag_b + mag_n, \quad (2.3)$$

Di mana:

$mag$  adalah jumlah medan magnet bumi dan medan magnet lainnya yang diukur dengan magnetometer.

$mag_r$  adalah jumlah medan magnet bumi dan medan magnet lainnya yang sebenarnya.

$mag_b$  adalah bias magnetometer.

$mag_n$  adalah *noise* magnetometer.

(Michel, Genevès, Fourati, & Layaïda, 2018)

### 2.1.3.3 Global Positioning System (GPS)

*Global Positioning System*, atau yang biasa disebut penerima GPS, adalah perangkat yang mampu menerima informasi dari satelit *Global Navigation Satellite System* dan kemudian menghitung posisi geografis perangkat. Dengan menggunakan perangkat lunak yang sesuai, perangkat dapat menampilkan posisinya di peta, dan menunjukkan arah rute.

*Global Navigation Satellite System* (GNSS) adalah Sistem navigasi satelit dengan cakupan global. Sistem navigasi adalah sistem yang menggunakan satelit untuk menyediakan pemosisian geo-spasial secara otomatis. Geo-spasial adalah aspek keruangan yang menunjukkan lokasi, letak, dan posisi suatu objek atau kejadian yang berada di bawah, pada, atau di atas permukaan bumi yang dinyatakan dalam sistem koordinat tertentu. Hal ini memungkinkan penerima elektronik kecil untuk menentukan lokasinya (bujur, lintang, dan ketinggian / elevasi) dengan presisi tinggi (dalam beberapa sentimeter hingga meter) menggunakan sinyal waktu yang ditransmisikan sepanjang garis pandang oleh radio dari satelit (Dewan Perwakilan Rakyat Republik Indonesia, 2011).

### 2.1.4 Transformasi

Transformasi adalah operasi yang mengambil entitas seperti titik, vektor, atau warna dan mengubahnya dengan cara tertentu. Transformasi sangat penting dalam grafika komputer. Transformasi dapat mengganti posisi maupun bentuk. Selain itu transformasi juga dapat menganimasikan objek, lampu, dan kamera. Transformasi juga dapat memastikan bahwa semua komputasi dilakukan dalam sistem koordinat yang sama, dan dapat memproyeksikan objek kepada bidang dengan berbagai macam cara (Akenine-Möller, Haines, & Hoffman, 2018).

#### 2.1.4.1 Translasi

Perubahan dari satu lokasi ke lokasi lain diwakili oleh matriks translasi ( $T$ ). Matriks ini me-translasi suatu entitas dengan vektor  $t = (t_x, t_y, t_z)$ .  $T$  diberikan di bawah ini

$$\mathbf{T}(\mathbf{t}) = \mathbf{T}(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Gambar 2.6 Matriks translasi

(Akenine-Möller, Haines, & Hoffman, 2018)

#### 2.1.4.2 Rotasi

Transformasi rotasi memutar suatu vektor (posisi atau arah) dengan sudut tertentu memutari sumbu tertentu yang melewati titik asal. Matriks rotasinya adalah  $R_x(\phi)$ ,  $R_y(\phi)$ , dan  $R_z(\phi)$ , yang memutar suatu entitas sebesar  $\phi$  radian di sekitar sumbu x, y, dan z, masing-masing.

$$\mathbf{R}_x(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi & 0 \\ 0 & \sin \phi & \cos \phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$\mathbf{R}_y(\phi) = \begin{pmatrix} \cos \phi & 0 & \sin \phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \phi & 0 & \cos \phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$\mathbf{R}_z(\phi) = \begin{pmatrix} \cos \phi & -\sin \phi & 0 & 0 \\ \sin \phi & \cos \phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Gambar 2.7 Matriks rotasi

(Akenine-Möller, Haines, & Hoffman, 2018)

#### 2.1.4.3 Dilatasi

Matriks dilatasi,  $S(s) = S(s_x, s_y, s_z)$ , mengganti skala entitas dengan faktor  $s_x$ ,  $s_y$ , dan  $s_z$  di sepanjang arah x, y, z masing-masing. Ini berarti bahwa matriks dilatasi dapat digunakan untuk memperbesar atau memperkecil suatu objek. Semakin besar  $s_i$ ,  $i \in \{x, y, z\}$ , semakin besar skala entitas ke arah tersebut. Menyetel salah satu komponen dari s ke 1 secara alami menghindari perubahan skala ke arah itu.

$$\mathbf{S}(\mathbf{s}) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Gambar 2.8 Matriks translasi

(Akenine-Möller, Haines, & Hoffman, 2018)

## 2.2 Tinjauan Studi

### 2.2.1 Markerless Augmented Reality with Light Source Estimation for Direct Illumination.

(Frahm, Koeser, Grest, & Koch, 2005)

- Masalah yang dibahas pada penelitian ini adalah sistem augmented reality tanpa penanda dengan pemosisian intuitif dan pencahayaan langsung objek virtual yang realistik.
- Metode yang digunakan pada penelitian ini adalah penggunaan kamera televisi untuk penangkapan video dunia nyata dan kamera fish-eye untuk melacak sumber cahaya. Bayangan dibuat dengan directional lighting. Cara kerja dari metode ini adalah penangkapan video menggunakan kamera televisi, lalu merekonstruksi dan memperkirakan posisi kedua kamera, setelah itu sinkronisasi antara dua kamera, sesudah sinkronisasi maka objek 3D virtual baru dapat dirender dan diperkirakan sumber cahayanya melalui kamera fish-eye, terakhir semuanya dirender dan ditambahkan pada dunia nyata menggunakan OpenSG yang berbasis OpenGL secara online.
- Hasil yang didapatkan pada penelitian ini adalah sistem augmented reality tanpa penanda yang mampu menghitung posisi sumber cahaya dan bayangan objek virtual pada bidang pemandangan 3D
- Kelemahan dari penelitian ini adalah pelacakan pada penelitian ini membutuhkan kamera lebih dari satu, lalu kamera yang dibutuhkan berbeda jenis, dan kamera fish-eye tidak dimiliki oleh kebanyakan orang. Kamera yang digunakan juga membutuhkan sinkronisasi, sehingga membutuhkan waktu setelah kamera digerakkan.

### 2.2.2 A tangible interface for 3D character animation using augmented reality technology.

(Pantuwong, 2016)

- Masalah yang dibahas pada penelitian ini adalah membuat teknik untuk berinteraksi dengan karakter 3D pada augmented reality menggunakan penanda.
- Metode yang digunakan pada penelitian ini adalah dengan menggunakan objek penanda berbentuk kubus, tablet, dan meja atau permukaan datar horizontal untuk berinteraksi dengan karakter 3D dengan cara mengarahkan kamera tablet ke objek penanda di atas meja lalu merender karakter 3D pada layar tablet. Objek penanda dapat diputar agar penanda yang dilacak berubah untuk mengganti animasi pada karakter 3D. Untuk pembuatan aplikasi, digunakan Unity Game Engine dengan library Vuforia dari Qualcomm untuk tujuan augmented reality.
- Hasil yang didapatkan pada penelitian ini adalah teknik interaksi yang dibuat pada penelitian ini lebih baik daripada interaksi pada game The Sims. Hasil didapatkan dari pengisian kuesioner oleh pengguna setelah pengguna memainkan game The Sims dan menggunakan aplikasi yang dibuat pada penelitian ini.
- Kelemahan dari penelitian ini adalah animasi yang dapat digunakan terbatas pada enam sisi kubus dan tidak ada parameter yang dapat diatur pada animasi, selain itu aplikasi harus menggunakan penanda khusus yang dibuat untuk menampilkan karakter 3D dan tidak ada metode untuk menyamakan pencahayaan pada karakter 3D dengan pencahayaan pada dunia nyata

### **2.2.3 An Innovative App With For Location Finding With Augmented Reality Using CLOUD.**

(Meenakshi, Vasudevan, Ritesh, & Santhosh, 2015)

- Masalah yang dibahas pada penelitian ini adalah membantu memahami bagaimana cara menggunakan teknologi cloud dalam pengembangan aplikasi augmented reality berbasis lokasi secara efisien.
- Metode yang digunakan pada penelitian ini adalah dengan menggunakan format XML untuk menyimpan data lokasi pada server yang informasinya dapat digunakan oleh aplikasi menggunakan plugin cloud dan GPS untuk pelacakan lokasi dengan cara membandingkan latitude, longitude, altitude (LLA).
- Hasil yang didapatkan pada penelitian ini adalah menyoroti penggunaan dari cloud pada web based Augmented Reality. Aplikasi Augmented Reality berbasis lokasi dapat memiliki pengaruh yang besar pada pasar karena menyediakan fungsi yang bermanfaat

- Kelemahan dari penelitian ini adalah aplikasi masih kurang interaktif dan belum dapat menambah lokasi yang ditampilkan, selain itu karena aplikasi yang digunakan berbasis web, maka dibutuhkan server yang selalu menyala untuk menyediakan informasi kepada aplikasi.

### **3. ANALISIS DAN DESAIN SISTEM**

Pada bab ini dibahas mengenai sistem yang dibuat, meliputi analisis permasalahan, desain sistem, dan desain aplikasi. Tahap desain dan perancangan sistem merupakan tahap awal yang dilakukan sebelum melakukan pemrograman. Hal ini bertujuan agar aplikasi yang dibuat dapat terstruktur dengan baik.

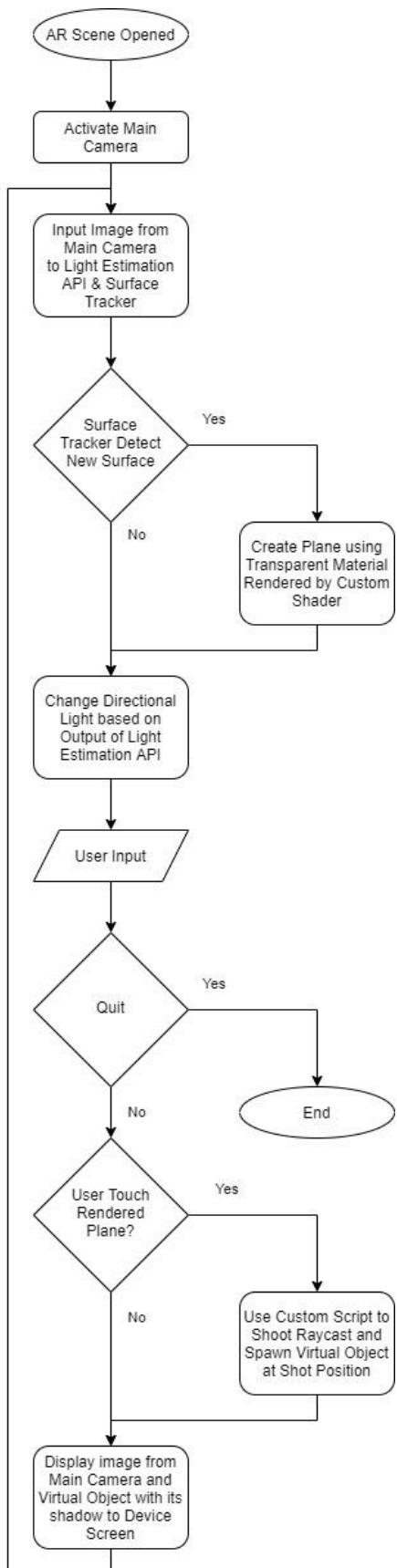
#### **3.1 Analysis Permasalahan**

Pada Subbab ini, akan dibahas analisis permasalahan yang muncul di pencahayaan terhadap *rendering* objek 3D virtual pada AR di perangkat *mobile*. Hal ini dilakukan dengan tujuan untuk mencari dan memberi solusi pemecahan masalah tersebut.

Berdasarkan tinjauan studi pada bab 2.2, dapat dilihat bahwa masalah-masalah yang masih ada pada aplikasi penelitian yang serupa adalah membutuhkan perangkat tambahan untuk menjalankan aplikasinya, seperti beberapa kamera khusus, penanda untuk mengeluarkan objek virtual, dan server untuk menyediakan data. Oleh karena itu, fitur-fitur yang dibutuhkan untuk menyelesaikan masalah pada aplikasi penelitian ini adalah fitur untuk mengganti animasi, transformasi, dan model tanpa menggunakan penanda, lalu fitur untuk menyimpan dan mengganti-ganti data pada internet langsung dari aplikasi tanpa perlu menyalakan server sendiri, dan fitur untuk mengestimasikan cahaya pada perangkat *mobile* umum.

Fitur untuk mengganti animasi, transformasi, dan model dapat dilakukan dengan cara menampilkan tombol untuk mengganti animasi pada *user interface* aplikasi, lalu fitur untuk data pada internet dapat dilakukan dengan menggunakan penyedia layanan *database cloud*.

*Light estimation* seperti yang sudah dijelaskan pada subbab 2.1.2.2 merupakan metode yang digunakan untuk mengestimasi pencahayaan pada dunia nyata agar informasinya dapat digunakan dalam pencahayaan dunia virtual. Pada penelitian ini, metode *light estimation* digunakan sebagai solusi untuk menyelesaikan masalah pencahayaan AR di perangkat *mobile*. Berikut ini adalah penggunaan metode *light estimation* pada aplikasi penelitian ini.

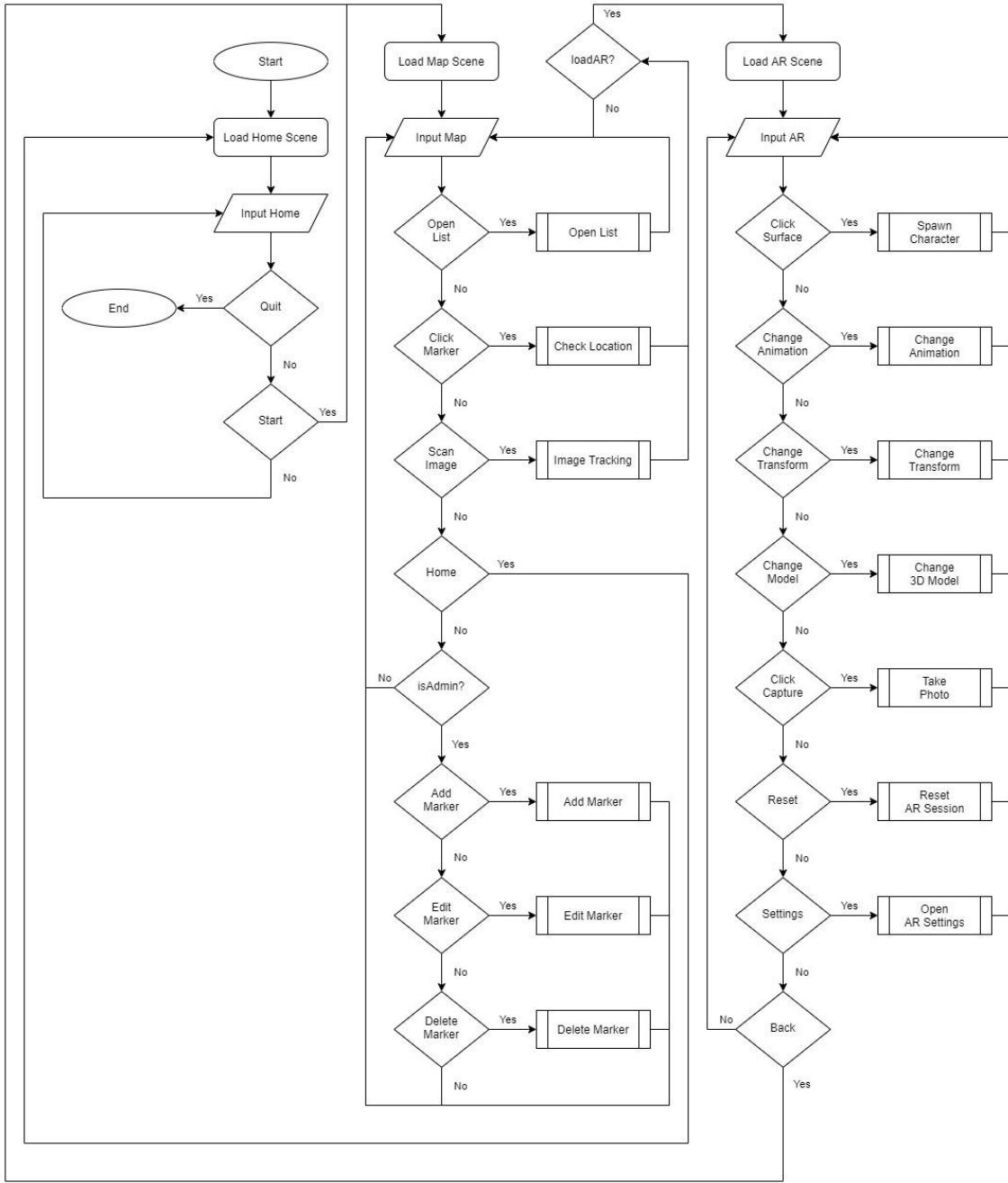


Gambar 3.1 Solusi masalah pencahayaan AR di perangkat mobile

Pada Gambar 3.1, dapat dilihat bahwa pada aplikasi penelitian ini, metode *light estimation* digunakan pada mode AR. Ketika adegan AR dibuka, kamera utama perangkat diaktifkan untuk membuat gambar yang digunakan sebagai *input* metode *light estimation API* dan *surface tracker* pada *library* ARCore, dan sebagai dasar *output* yang ditampilkan pada layar perangkat. *Surface tracker* kemudian membuat bidang permukaan pada dunia virtual sesuai dengan permukaan pada dunia nyata, yang menggunakan material transparan yang *dirender* oleh *shader* khusus agar dapat menerima bayangan. Objek 3D virtual kemudian dapat dimunculkan dengan cara menggunakan *script* untuk menembak *raycast* pada bidang permukaan yang dihasilkan dan mengeluarkannya pada posisi hasil tembakan. Hasil dari *light estimation API* berupa informasi cahaya digunakan untuk mengatur pencahayaan pada dunia virtual agar bayangan yang dihasilkan oleh objek 3D virtual sesuai dengan bayangan yang ada pada dunia nyata. Objek dan bidang transparan dengan bayangan yang dihasilkan ini kemudian ditambahkan pada gambar yang dihasilkan kamera utama untuk ditampilkan sebagai *output* pada layar perangkat.

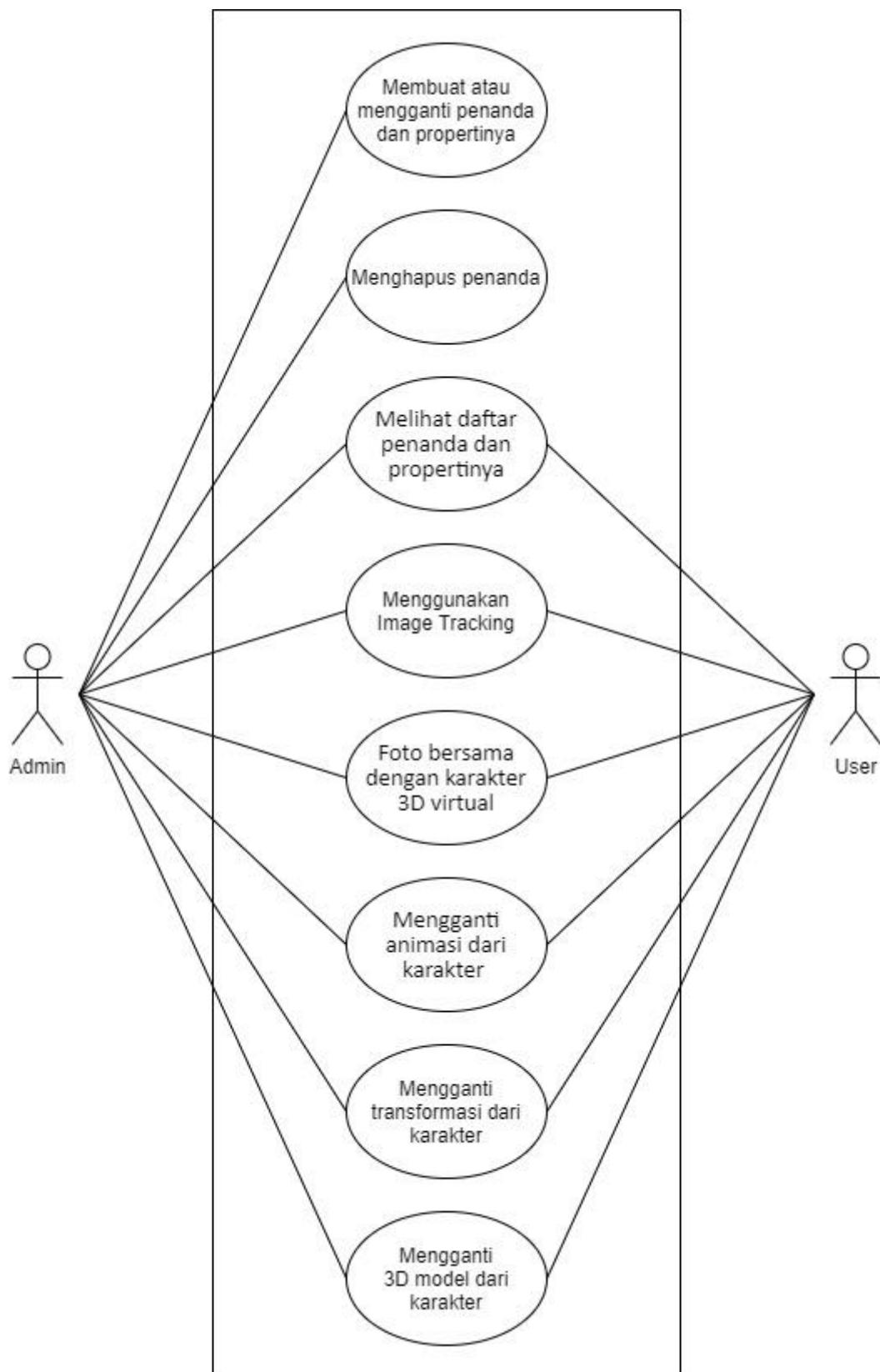
### 3.2 Desain Sistem

Dalam pembuatan aplikasi penelitian ini, desain sistem berupa garis besar sistem dapat dibuat dalam bentuk flowchart berdasarkan analisis permasalahan yang dibahas pada subbab sebelum ini. Flowchart garis besar proses dapat dilihat pada gambar 3.2, dan *Use Case Diagram* dapat dilihat pada gambar 3.3.



Gambar 3.2 Flowchart garis besar sistem secara keseluruhan

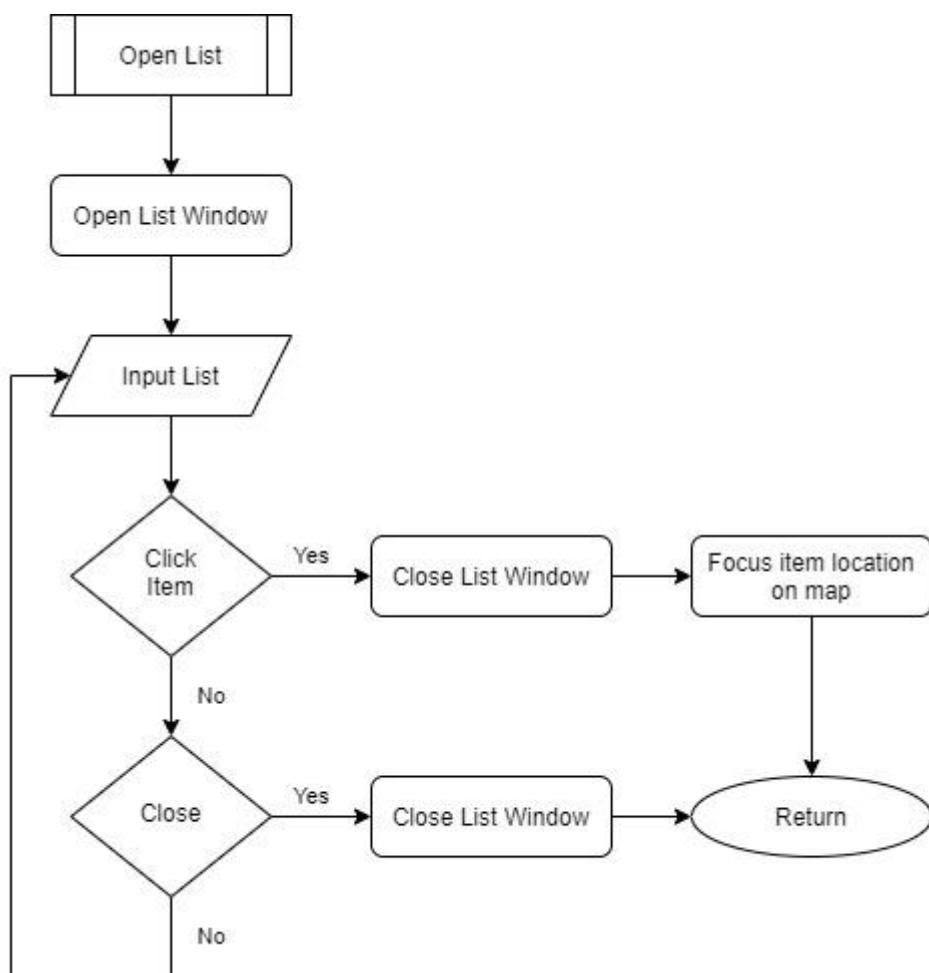
Aplikasi dibagi menjadi dua versi, yaitu untuk admin, dan untuk pengguna biasa. Aplikasi dimulai dengan memuat adegan awal. Pada adegan ini pengguna dapat memilih untuk keluar dari aplikasi atau melanjutkan ke adegan peta. Setelah pengguna memilih untuk melanjutkan dan adegan peta selesai dimuat, pengguna dapat memilih untuk memilih untuk membuka daftar penanda, memencet atau berjalan ke penanda, menuju adegan *image tracking*, atau kembali ke adegan awal. Jika pengguna adalah admin, pengguna juga dapat menambah penanda, mengedit penanda, dan menghapus penanda. Berikut adalah *Use Case Diagram* dari aplikasi.



Gambar 3.3 Use Case Diagram

Pada adegan AR, pengguna dapat memilih untuk memencet permukaan, mengganti animasi, mengganti transformasi, mengganti model, memencet tombol ambil foto, melakukan reset sesi AR, membuka panel *settings*, dan kembali ke adegan peta. Penjelasan lebih lengkap tentang masing-masing proses pada flowchart garis besar sistem secara keseluruhan akan dijelaskan pada bagian berikut.

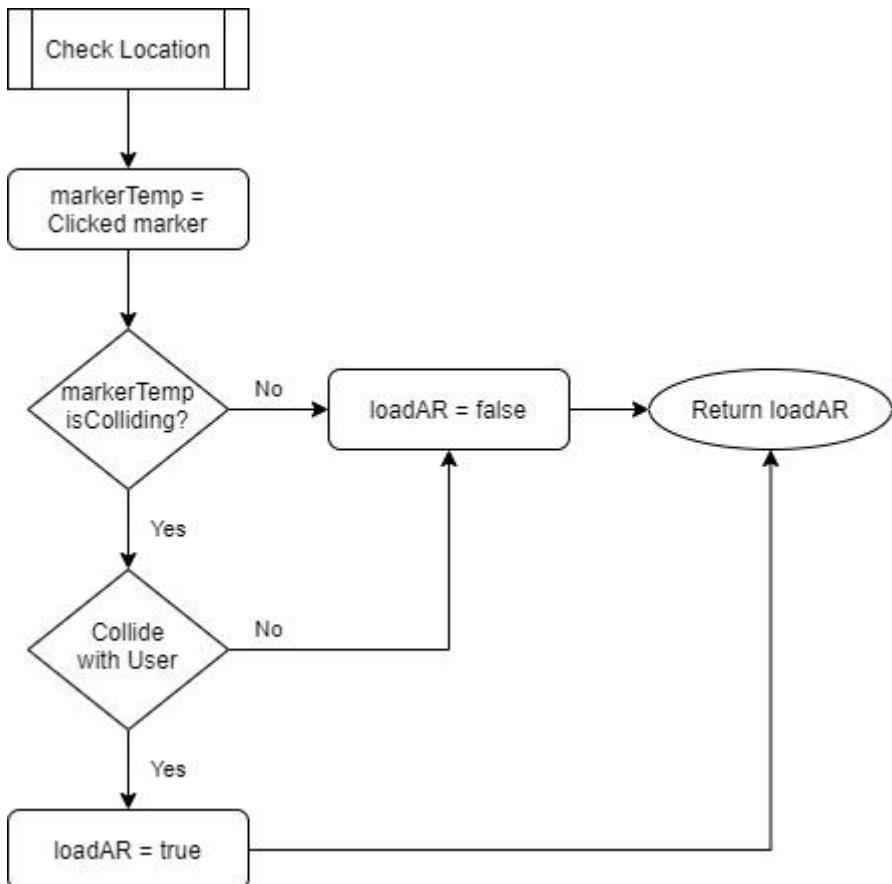
- Proses membuka daftar penanda



Gambar 3.4 Flowchart buka daftar penanda

Pada proses membuka daftar penanda (gambar 3.4), pengguna dapat memilih untuk menuju lokasi item yang dipilih pada peta, dan dapat menutup daftar penanda.

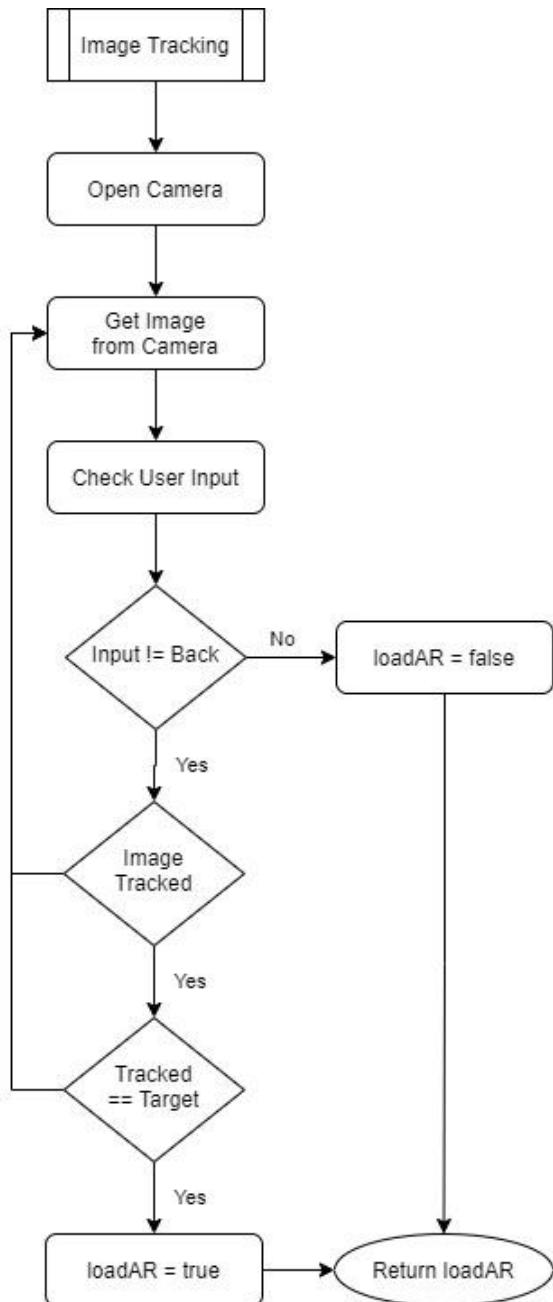
- Proses mengecek lokasi



Gambar 3.5 Flowchart pengecekan lokasi

Proses mengecek lokasi seperti yang dapat dilihat pada gambar 3.5, dimulai dengan pengecekan penanda yang dipencet pada peta, apakah bertabrakan atau tidak, lalu di cek apakah benda yang bertabrakan dengan penanda tersebut adalah pengguna. Jika penanda yang dipencet tidak bertabrakan dengan pengguna, maka variabel “*loadAR*” akan diatur menjadi *false*, sehingga tidak terjadi apa-apa, tetapi jika penanda yang dipencet bertabrakan dengan pengguna, maka variabel “*loadAR*” akan diatur menjadi *true*, sehingga dapat memuat dan berpindah ke adegan AR.

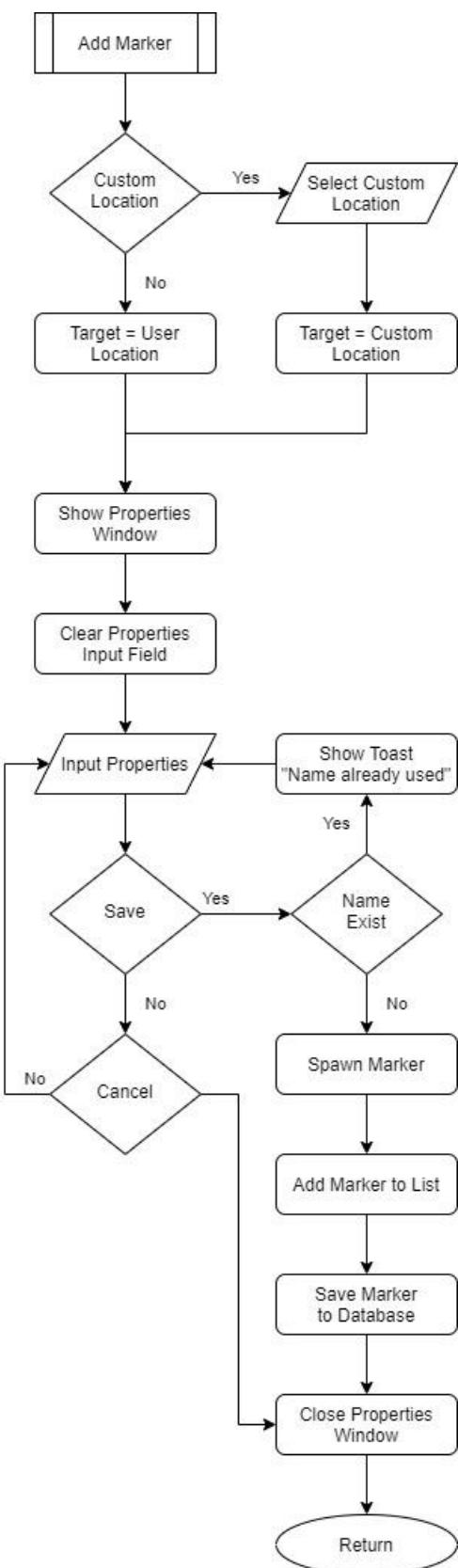
- Proses melacak gambar



Gambar 3.6 Flowchart pelacakan gambar

Proses melacak gambar pada gambar 3.6 dimulai dengan membuka kamera utama pada perangkat sebagai sumber gambar yang digunakan untuk *input* pelacakan gambar. Setelah itu, pada proses ini pengguna dapat menutup pelacakan gambar dengan memencet tombol kembali. Pengguna juga dapat mengarahkan kamera ke target penanda yang ditentukan untuk memuat dan berpindah ke adegan AR.

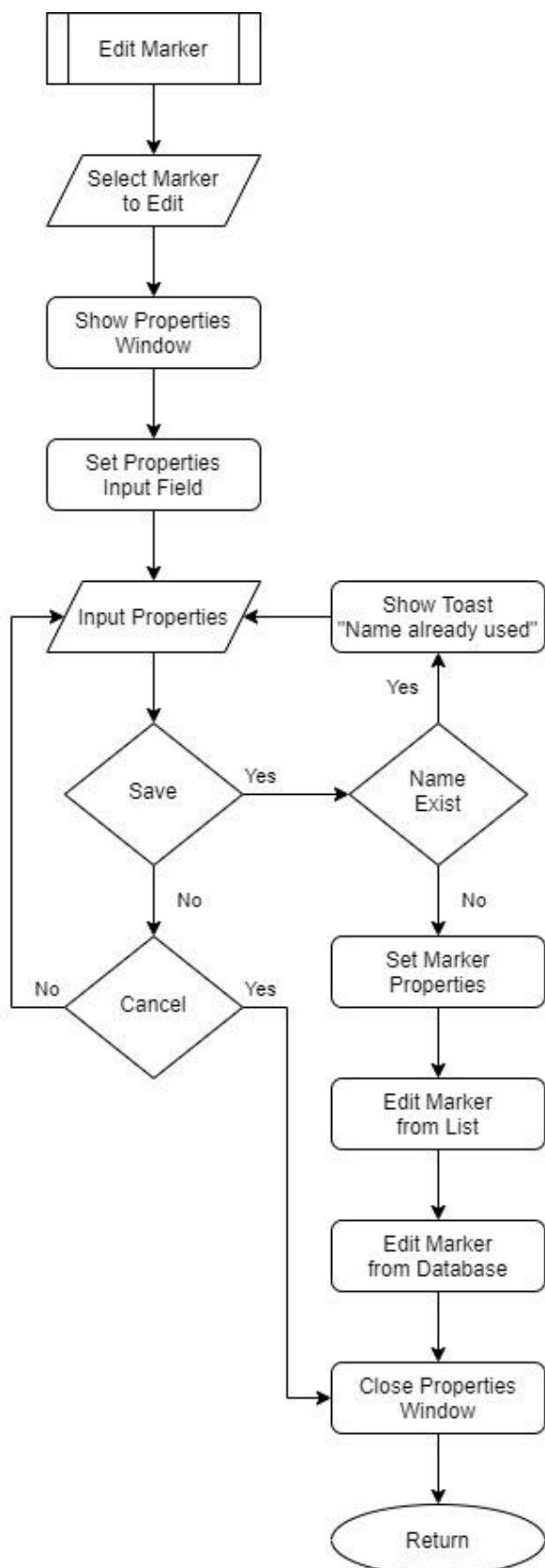
- Proses menambah penanda



Gambar 3. 7 Flowchart tambah penanda

Pada gambar 3.7, dapat dilihat proses untuk menambah penanda. Proses menambah penanda dimulai dengan mengecek apakah tombol yang dipencet adalah tombol untuk menambah penanda pada lokasi pengguna ataupun lokasi tertentu. Jika tombol yang dipencet adalah tombol untuk menambah penanda pada lokasi tertentu, pengguna dapat memilih lokasi pada peta sebagai target lokasi dikeluarkannya penanda. Setelah menentukan target lokasi, selanjutnya ditampilkan panel untuk mengisi informasi penanda. Pada panel informasi, pengguna dapat memilih untuk menyimpan ataupun membatalkan penambahan penanda. Jika pengguna memilih untuk menyimpan penanda, maka penanda akan dikeluarkan pada target lokasi, lalu informasinya ditambahkan pada daftar penanda dan *database*.

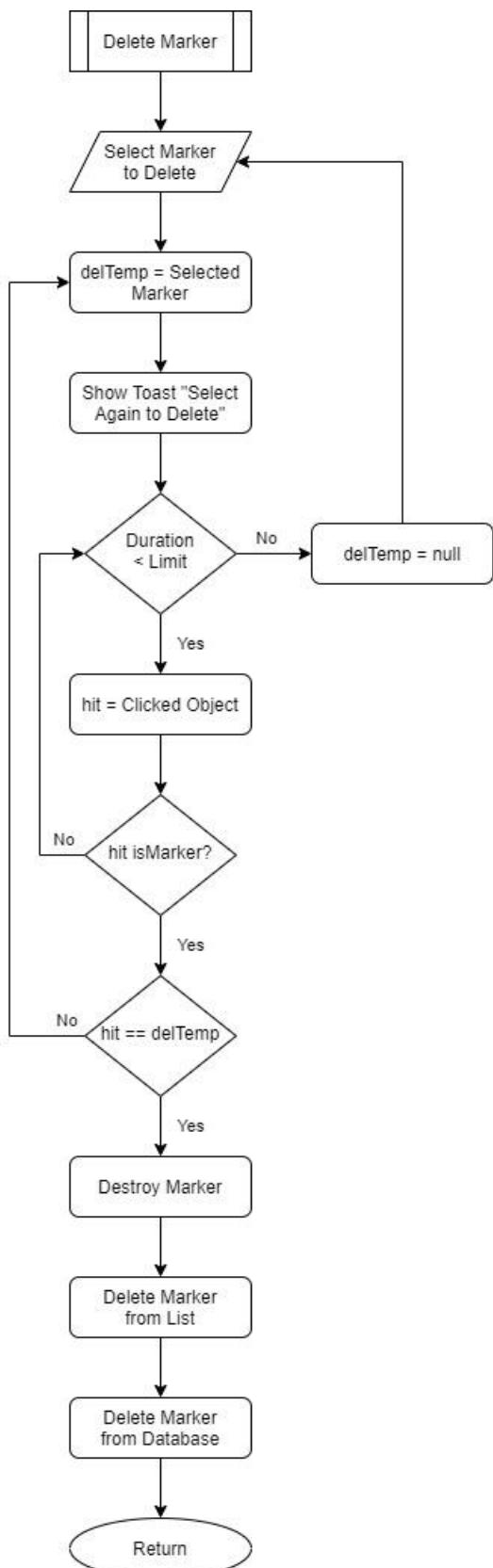
- Proses mengedit penanda



Gambar 3.8 Flowchart edit penanda

Pada gambar 3.8, dapat dilihat proses untuk mengedit penanda. Proses mengedit penanda dimulai dengan menunggu *input* dari pengguna untuk memilih penanda pada peta sebagai target penanda yang akan di edit. Setelah menentukan target penanda, selanjutnya ditampilkan panel untuk mengganti informasi penanda. Pada panel informasi, pengguna dapat memilih untuk menyimpan ataupun membatalkan pengeditan penanda. Jika pengguna memilih untuk menyimpan penanda, maka penanda akan dikeluarkan pada target lokasi, lalu informasinya diedit pada daftar penanda dan database.

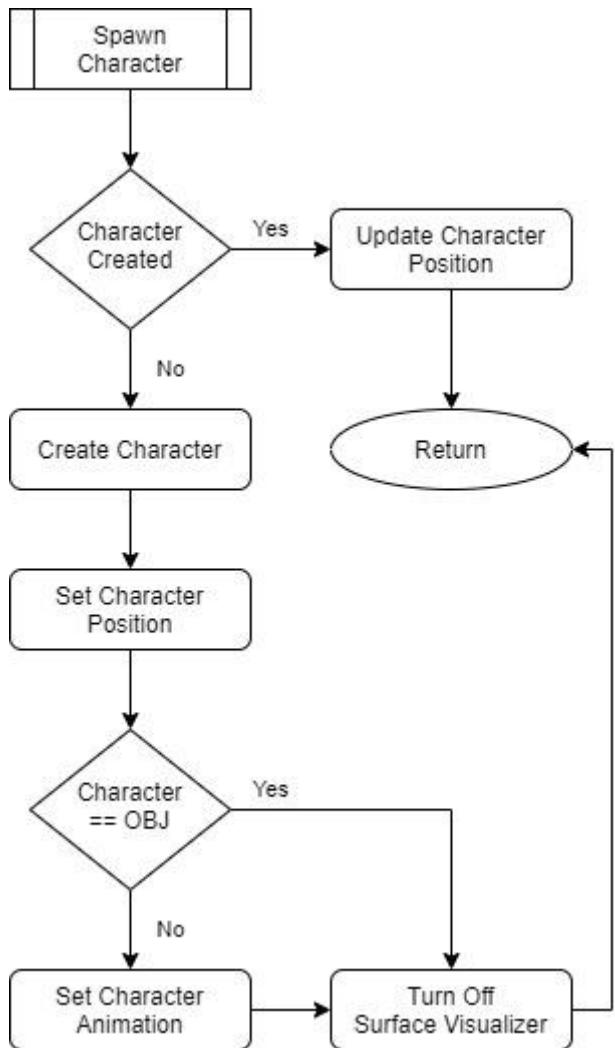
- Proses menghapus penanda



Gambar 3.9 Flowchart hapus penanda

Pada gambar 3.9, dapat dilihat proses untuk menghapus penanda. Proses menghapus penanda dimulai dengan menunggu *input* dari pengguna untuk memilih penanda pada peta sebagai target penanda yang akan dihapus. Setelah itu akan ditampilkan tulisan pencet lagi untuk menghapus, jika durasi tulisan ditampilkan telah habis dan tidak ada penanda yang dipilih maka akan kembali ke *input* awal. Jika penanda yang dipilih berbeda maka target penanda diganti menjadi penanda yang baru dipilih, tetapi jika penanda yang dipilih sama maka penanda akan dihancurkan, lalu dihapus dari daftar penanda dan *database*.

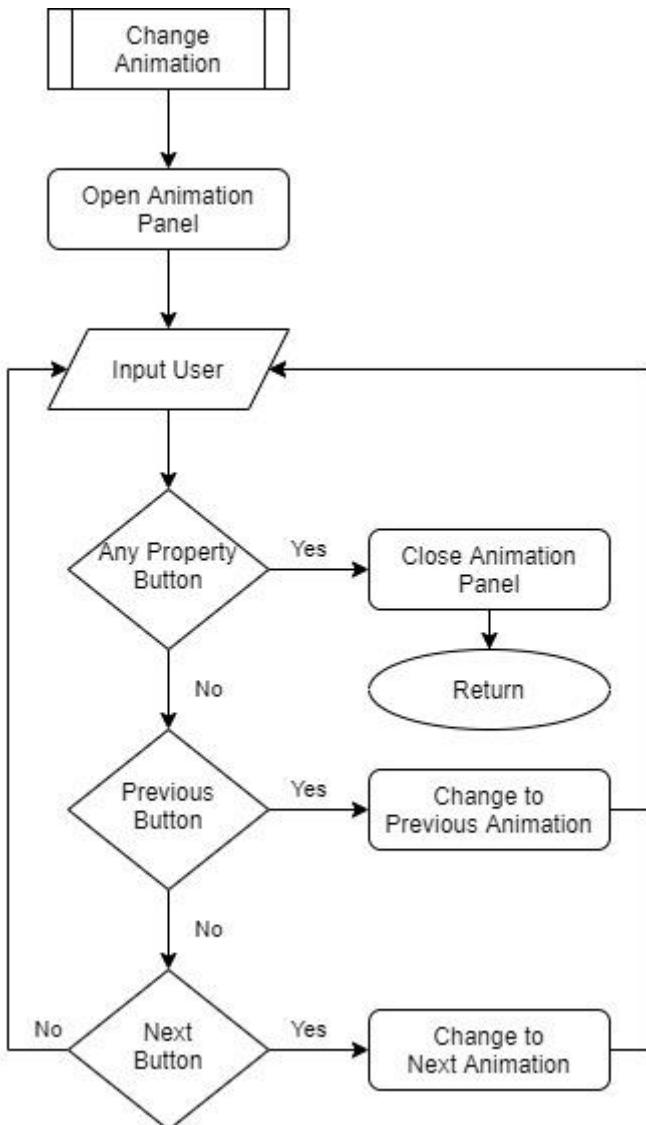
- Proses mengeluarkan karakter



Gambar 3.10 Flowchart *spawn character*

Proses mengeluarkan karakter pada gambar 3.10 dimulai dengan mengecek apakah sudah ada karakter yang dikeluarkan, jika sudah ada, maka yang dilakukan hanya memindah posisi karakter ke permukaan yang dipencet oleh pengguna. Jika karakter belum dikeluarkan maka setelah mengeluarkan karakter pada posisi permukaan yang dipencet oleh pengguna, dicek apakah karakter tersebut karakter hasil *import* atau tidak, jika tidak maka animasi karakter akan diset menjadi animasi dasar. Setelah itu *surface visualizer* akan dimatikan.

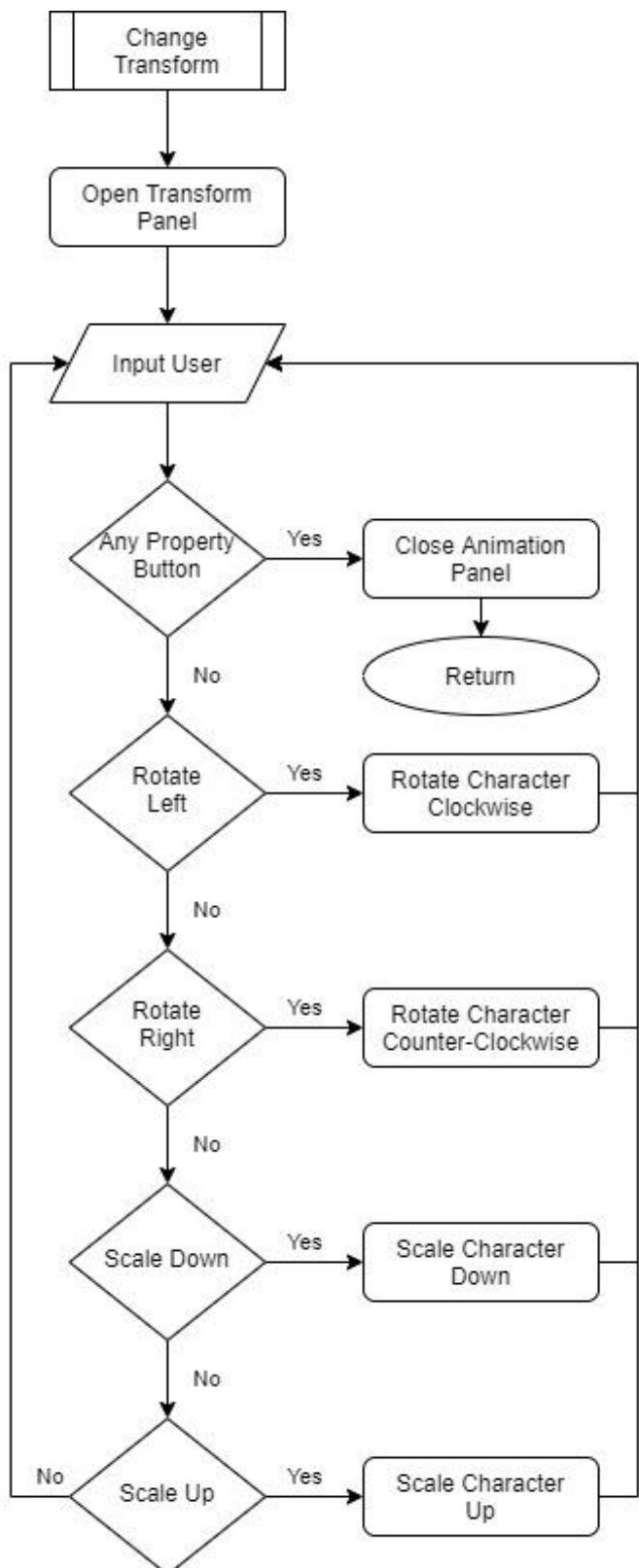
- Proses mengganti animasi karakter



Gambar 3.11 Flowchart ganti animasi

Pada proses untuk mengganti animasi seperti yang dapat di lihat pada gambar 3.11, proses dimulai dengan membuka panel animasi, lalu setelah itu menunggu *input* dari pengguna. Jika *input* dari pengguna merupakan salah satu tombol properti (Animasi, transformasi, model), maka panel akan ditutup dan proses diakhiri. Jika tombol untuk ganti animasi ke sebelum atau setelahnya dipencet, maka animasi karakter yang sedang aktif akan diganti menjadi animasi yang sesuai.

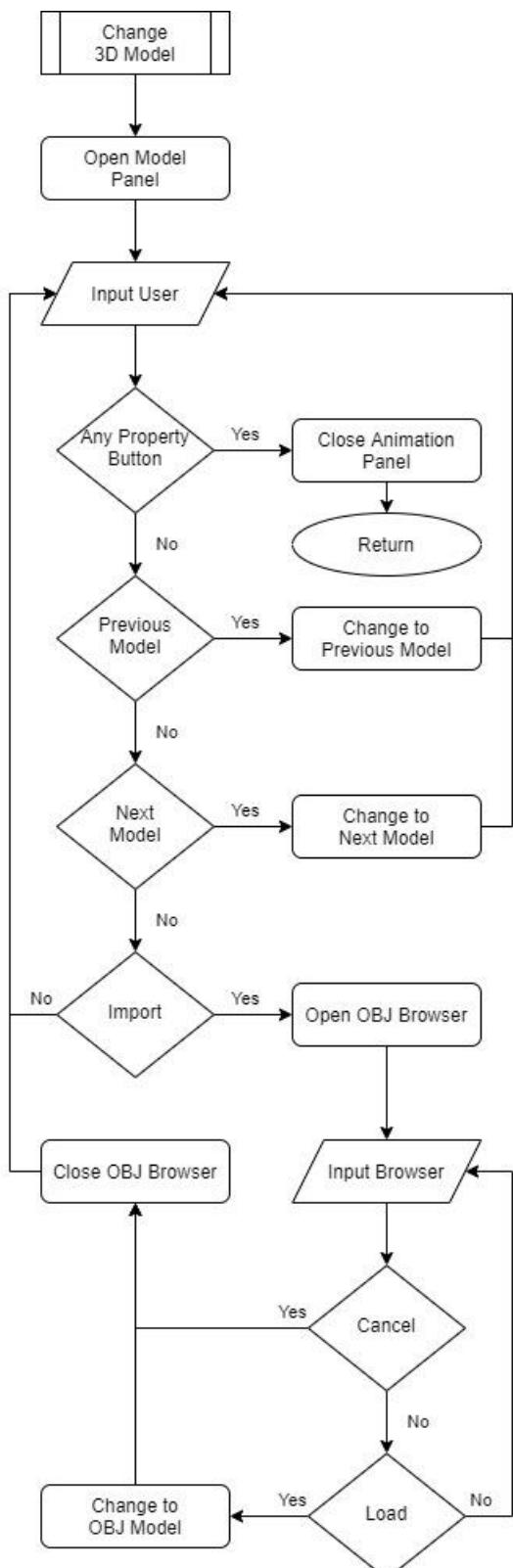
- Proses mengganti transformasi karakter



Gambar 3.12 Flowchart ganti transformasi

Pada proses untuk mengganti transformasi seperti yang dapat di lihat pada gambar 3.12, proses dimulai dengan membuka panel transformasi, lalu setelah itu menunggu *input* dari pengguna. Jika *input* dari pengguna merupakan salah satu tombol properti (Animasi, transformasi, model), maka panel akan ditutup dan proses diakhiri. Jika tombol untuk ganti rotasi ke kiri dipencet, maka karakter yang sedang aktif akan dirotasi searah jarum jam. Lalu jika tombol rotasi ke kanan yang dipencet, maka akan dirotasi berlawanan arah jarum jam. Jika tombol untuk membesarkan atau mengecilkan dipencet, maka skala karakter yang sedang aktif akan diganti menjadi semakin besar atau semakin kecil sesuai dengan tombol yang dipencet.

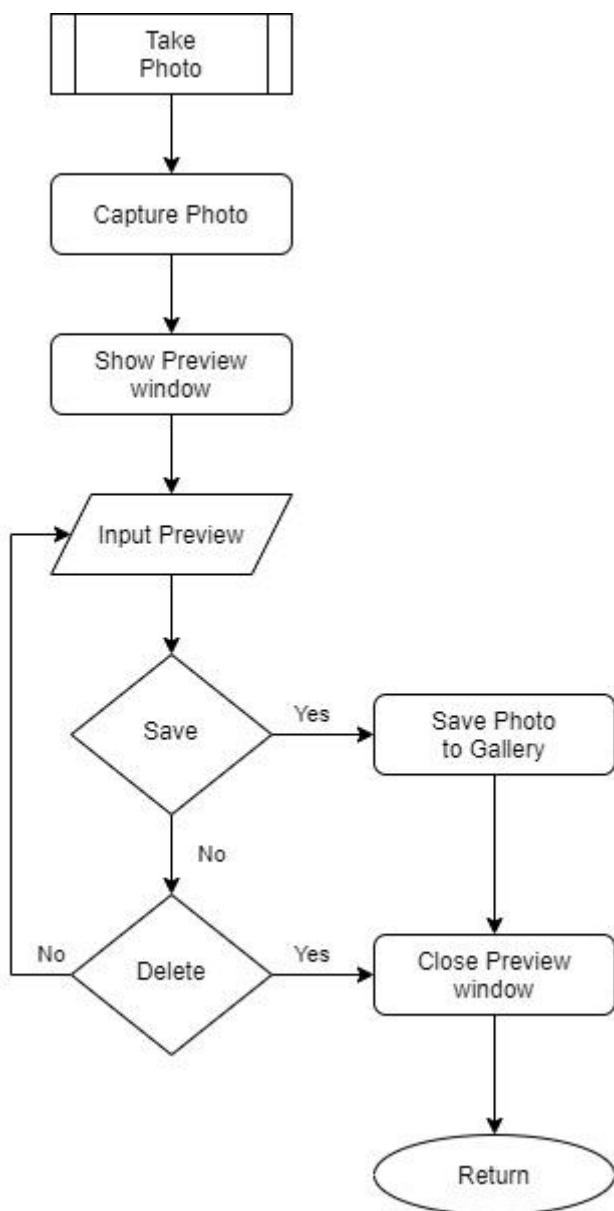
- Proses mengganti model 3D karakter



Gambar 3.13 Flowchart ganti model

Pada proses untuk mengganti model seperti yang dapat di lihat pada gambar 3.13, proses dimulai dengan membuka panel model, lalu setelah itu menunggu *input* dari pengguna. Jika *input* dari pengguna merupakan salah satu tombol properti (Animasi, transformasi, model), maka panel akan ditutup dan proses diakhiri. Jika tombol untuk ganti model ke sebelum atau setelahnya dipencet, maka karakter yang sedang aktif akan diganti sesuai tombol yang dipencet. Lalu jika tombol *import* yang dipencet, maka akan dibuka panel browser OBJ, setelah itu akan ditunggu *input* dari pengguna untuk memilih *file* yang akan digunakan. Jika pengguna memilih untuk memuat, maka model akan diganti menjadi model OBJ yang dimuat. Setelah itu panel browser OBJ akan ditutup dan kembali ke *input* pertama.

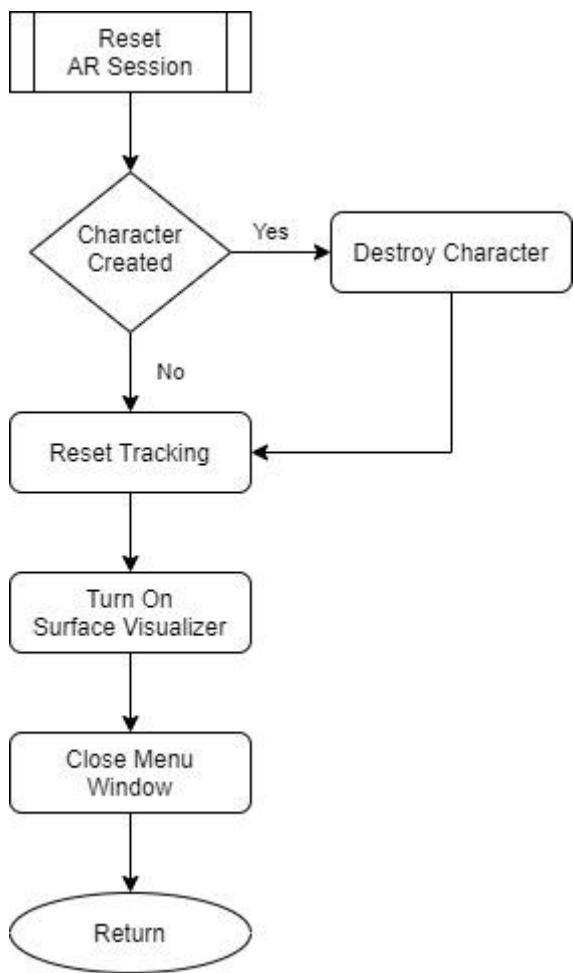
- Proses mengambil foto



Gambar 3.14 Flowchart ambil foto

Pengambilan foto pada gambar 3.14 dimulai dengan menangkap hasil foto dan menunjukkannya ke panel *preview*. Setelah itu pengguna dapat memilih untuk menyimpan ataupun menghapus hasil foto yang ditampilkan, jika pengguna memilih untuk menyimpannya, maka foto akan dipindahkan ke *gallery* dari perangkat pengguna. Setelah itu panel *preview* ditutup dan proses diakhiri.

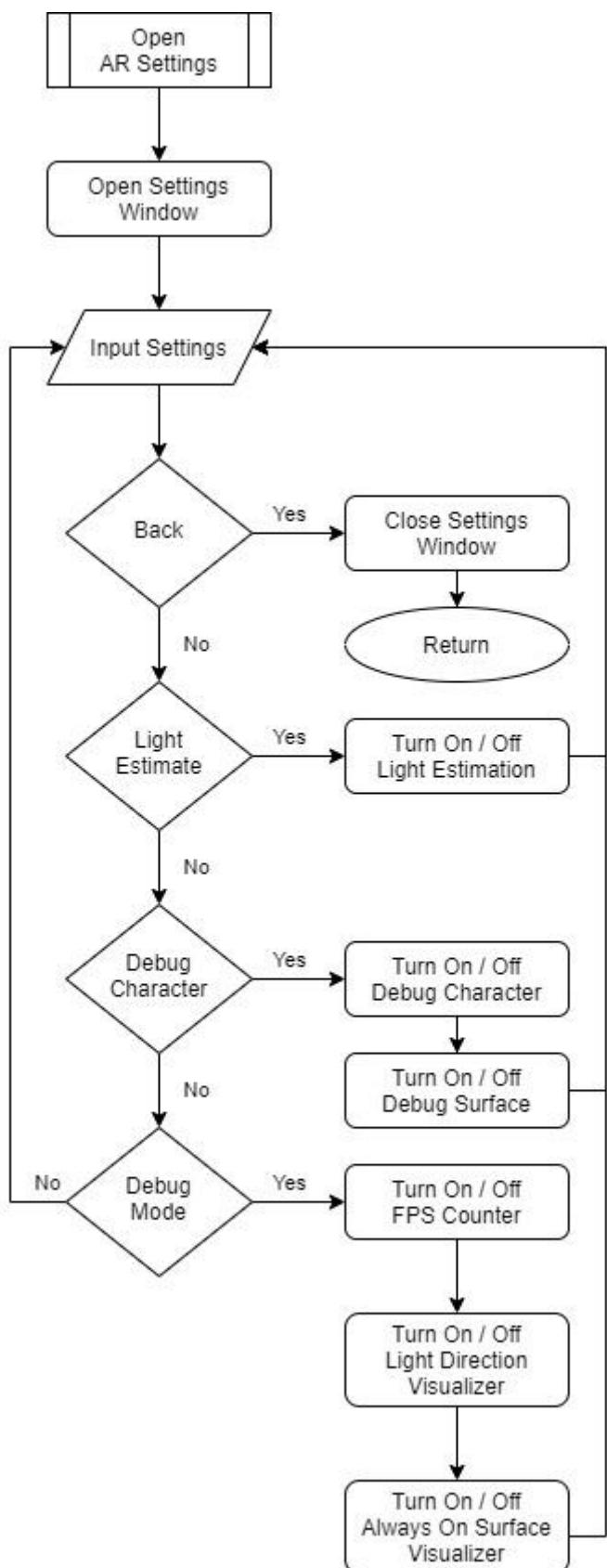
- Proses melakukan reset sesi AR



Gambar 3.15 Flowchart reset sesi

Reset sesi AR pada gambar 3.15 dimulai dengan mengecek apakah ada karakter yang sudah dibuat, apabila ada karakter yang sudah dibuat maka karakter tersebut dihancurkan. Setelah itu pelacakan di reset, lalu *surface visualizer* kembali dinyalakan. Setelah itu panel menu ditutup dan proses diakhiri.

- Proses membuka pengaturan mode AR



Gambar 3.16 Flowchart buka *settings*

Pembukaan pengaturan mode AR pada gambar 3.16 dimulai dengan pembukaan panel pengaturan, lalu setelah itu menunggu *input* dari pengguna. Pengguna dapat memencet tombol kembali untuk menutup panel pengaturan, dapat menyalakan atau mematikan metode *light estimation*, menyalakan atau mematikan karakter debug, dan menyalakan atau mematikan mode debug.

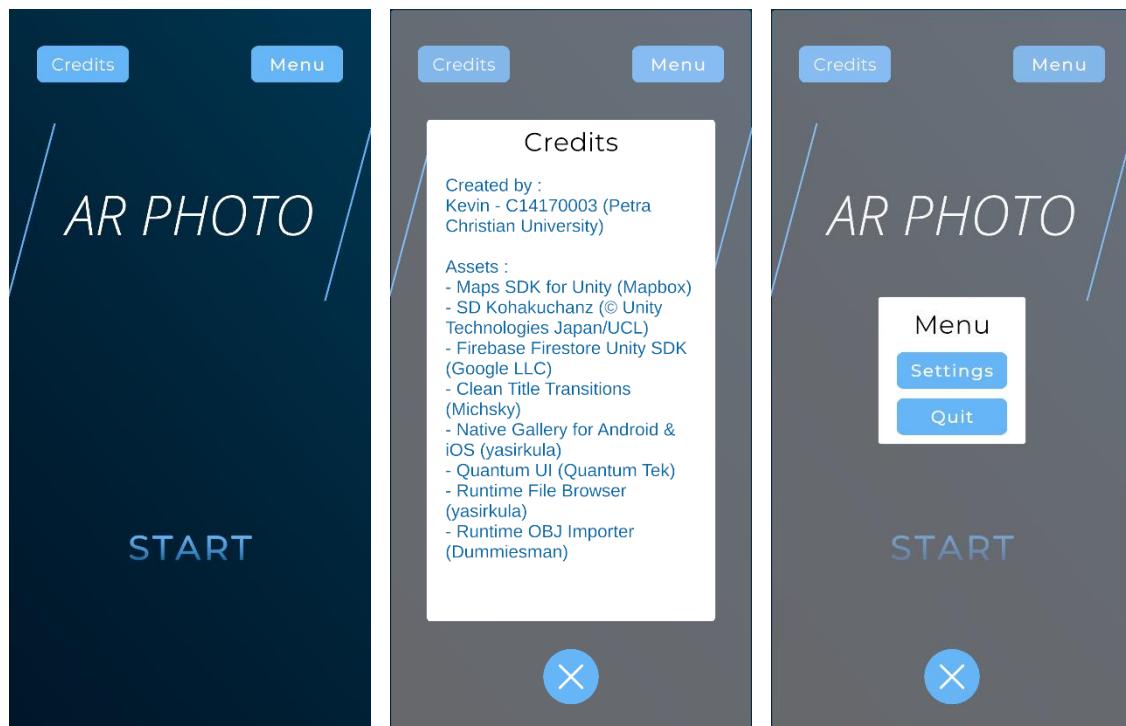
### 3.3 Desain Aplikasi

#### 3.3.1 Desain User Interface

Secara Umum, sistem perangkat lunak yang dikembangkan pada penelitian ini akan terdiri dari empat adegan utama, yaitu adegan awal, adegan peta, adegan *image tracking*, dan adegan AR.

##### 3.3.1.1 Desain Adegan Awal

UI yang ditampilkan pada gambar 3.17 adalah adegan awal, yaitu adegan yang pertama kali ditampilkan saat aplikasi dijalankan. Terdapat tiga hal yang dapat dilakukan pada adegan ini, yaitu start (memuat adegan peta), kredit (membuka panel kredit), dan menu (membuka panel menu).

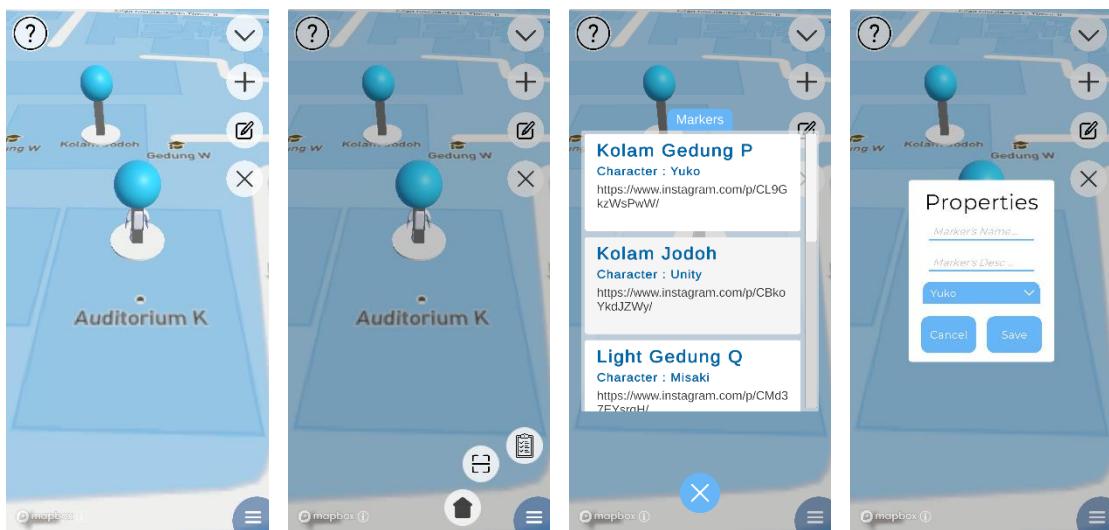


Gambar 3.17 UI adegan awal

Panel kredit berisi nama pembuat dan aset-aset yang digunakan pada aplikasi penelitian ini, lalu pada panel menu berisi tombol untuk keluar dari aplikasi dan tombol ke panel pengaturan yang berisi tombol untuk mematikan mode admin.

### 3.3.1.2 Desain Adegan Peta

Adegan peta pada gambar 3.18 adalah adegan untuk menampilkan lokasi pengguna pada peta. Pada adegan peta, pengguna dapat membuka menu pada bagian bawah kanan, yang berisi (dari kanan ke kiri) tombol untuk melihat panel daftar penanda, tombol untuk membuka adegan *image tracking*, dan tombol untuk kembali ke adegan awal. Pengguna dapat berpindah ke adegan AR jika lokasi pengguna berjalan masuk ke bagian putih penanda, memencet penanda tersebut, ataupun melalui adegan *image tracking*.

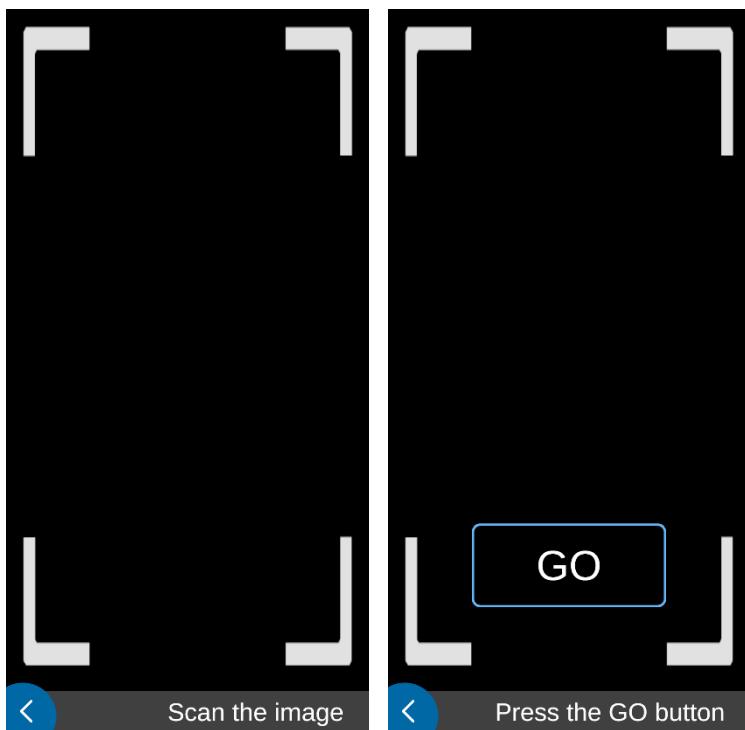


Gambar 3.18 UI adegan peta

Pada panel daftar penanda, pengguna dapat memencet salah satu item pada daftar penanda untuk menunjukkan lokasi item tersebut pada peta. Jika pengguna adalah admin, maka ditampilkan empat tombol pada bagian atas kanan untuk mengatur penanda, yaitu (dari atas ke bawah) tambah penanda pada lokasi pengguna, tambah penanda pada lokasi yang dipilih, edit informasi (nama, deskripsi, dan model karakter) penanda, dan hapus penanda.

### 3.3.1.3 Desain Adegan *Image Tracking*

Pada adegan *image tracking* (gambar 3.19), pengguna dapat melacak gambar sebagai pelacakan cadangan untuk menuju ke adegan AR dengan karakter acak, dan dapat kembali ke adegan peta. Untuk menggunakan pelacakan gambar, pengguna harus mengarahkan kamera pada perangkat agar gambar yang dilacak berada pada layar perangkat.



Gambar 3.19 UI adegan *image tracking*

Gambar yang digunakan untuk pelacakan gambar adalah gambar logo Universitas Kristen Petra (UKP) pada gambar 3.20. Setelah pengguna berhasil melacak gambar logo UKP, tombol untuk berpindah ke adegan AR akan ditampilkan.



Gambar 3.20 Logo UKP

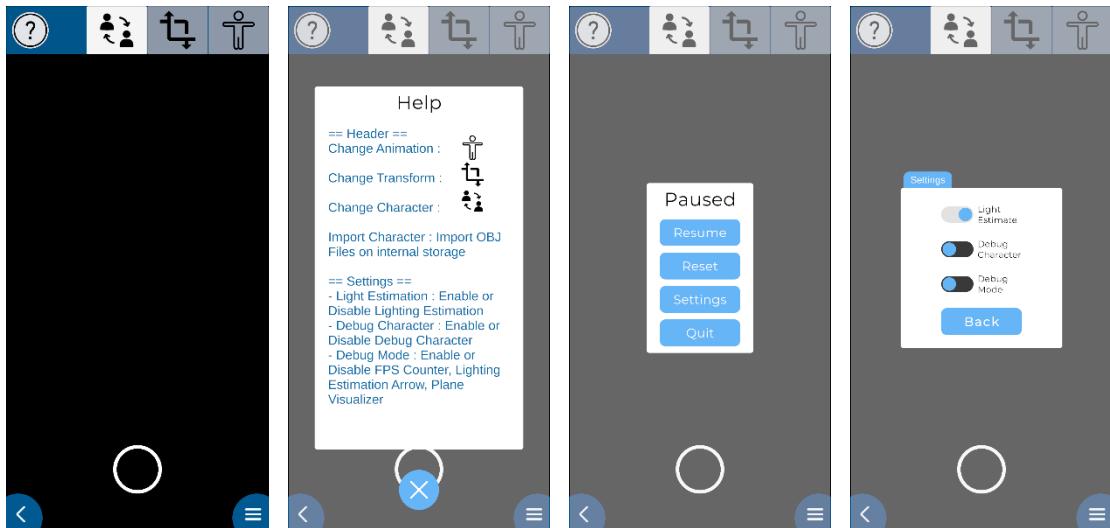
Sumber : *Petra Christian University*. (n.d.).

[https://en.wikipedia.org/wiki/Petra\\_Christian\\_University](https://en.wikipedia.org/wiki/Petra_Christian_University)

#### 3.3.1.4 Desain Adegan AR

Pada desain adegan AR di gambar 3.21, terdapat tombol bantuan di bagian atas kiri yang membuka panel bantuan yang berisi penjelasan adegan AR. Pada bagian bawah kiri terdapat tombol kembali ke adegan peta, lalu pada bagian bawah kanan terdapat tombol untuk

membuka panel menu. Panel menu berisi tombol untuk menutup menu, tombol untuk melakukan reset sesi AR, tombol untuk membuka panel pengaturan, dan tombol untuk keluar dari aplikasi, selain itu, panel menu juga membuat status aplikasi menjadi *paused*. Panel pengaturan berisi tombol untuk menyalakan atau mematikan metode *light estimation*, karakter debug, dan mode debug.



Gambar 3.21 UI adegan AR

Pada gambar 3.22, pengaturan karakter debug mengaktifkan karakter dan bidang permukaan yang selalu aktif, dan mode debug mengaktifkan penghitung *Frames Per Second*, penanda arah cahaya, dan *plane visualizer* yang selalu aktif. Pada bagian atas adegan AR, terdapat (dari kanan ke kiri) tombol untuk membuka panel animasi, panel transformasi, dan panel model.



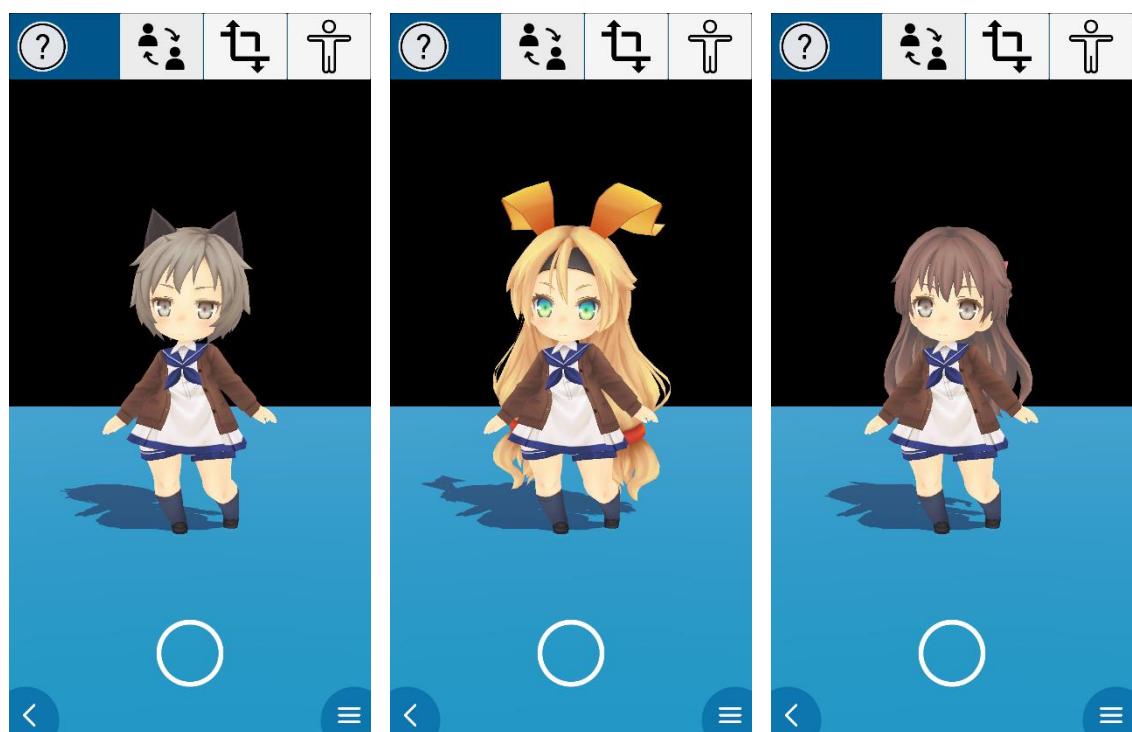
Gambar 3.22 Header adegan AR dan preview foto

Panel animasi berisi tombol untuk mengganti animasi pada suatu siklus (berdiri, berjalan, berlari, melompat berulang, terserang berulang, jatuh dan bangkit, pose kekalahan, pose kemenangan, melompat sekali). Panel transformasi berisi tombol untuk mengganti skala, dan tombol untuk rotasi karakter pada sumbu vertikal (sumbu Y atau yaw). Panel model berisi tombol untuk mengganti model pada suatu siklus (Yuko, Unity, Misaki, Yuko (Blue Alt), Misaki (Blue Alt), Yuko (Red Alt), Misaki (Red Alt)), dan tombol untuk *import file* dengan ekstensi OBJ pada penyimpanan internal perangkat untuk digunakan sebagai model 3D tanpa animasi.

Jika pengguna menekan tombol ambil foto, maka foto akan diambil dan ditampilkan pada panel *preview*. Pada panel *preview*, pengguna dapat memilih untuk menyimpan atau menghapus hasil foto yang diambil.

### 3.3.2 Desain Model 3D Karakter Virtual

Model 3D Karakter virtual pada gambar 3.23 adalah model yang digunakan pada adegan AR sebagai karakter yang dapat di munculkan oleh pengguna aplikasi, yang akan *dirender* secara *real-time*.



Gambar 3.23 Model 3D karakter virtual

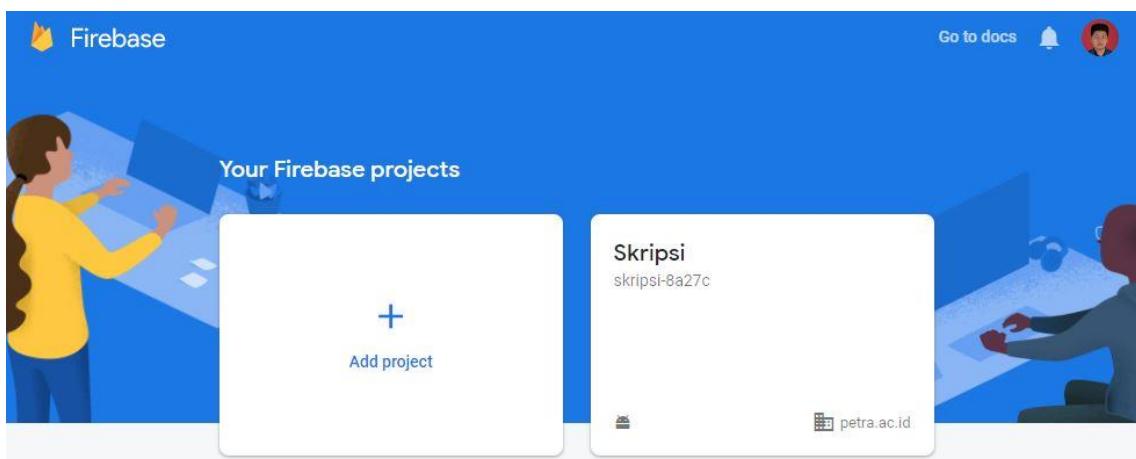
Desain model 3D karakter virtual didapatkan dari halaman mengunduh pada web resmi Unity-chan (<https://unity-chan.com/download/releaseNote.php?id=SDKohakuchanz&lang=en>).

## 4. IMPLEMENTASI SISTEM

Pada bab ini dijelaskan tentang implementasi sistem yang siap untuk digunakan sesuai dengan teori-teori dan desain sistem yang telah dibahas pada bab-bab sebelumnya. Implementasi sistem yang dibahas meliputi pembuatan *database* untuk aplikasi android yang digunakan dan implementasi aplikasinya sendiri beserta dengan potongan / segmen *source code* yang dibuat dengan menggunakan bahasa pemrograman C Sharp (C#).

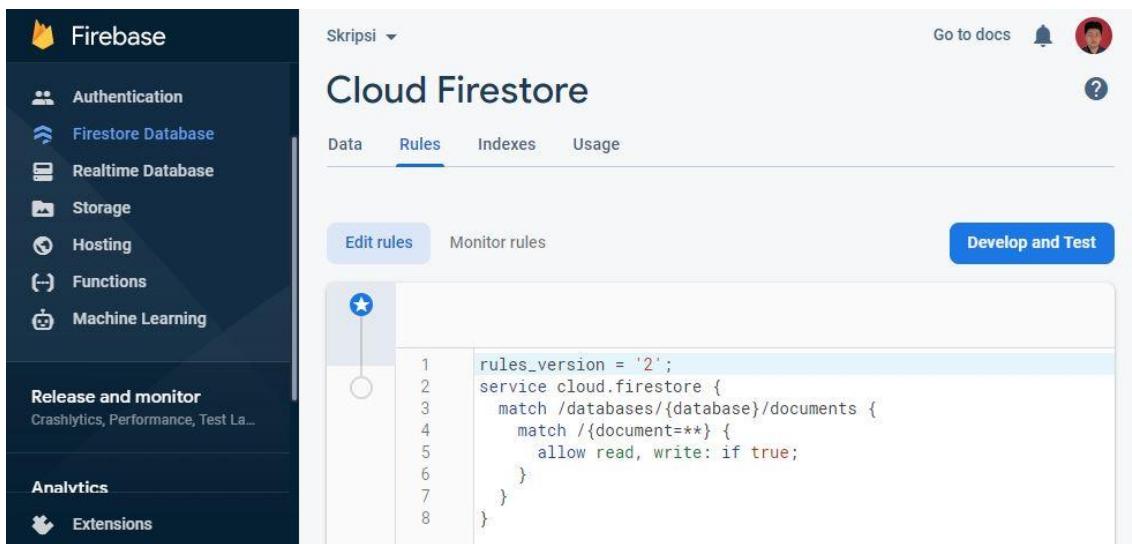
### 4.1 Pembuatan *Database* untuk Aplikasi Android yang digunakan

Pada pembuatan dan implementasi sistem yang digunakan, salah satu hal yang diperlukan adalah pembuatan *database* untuk aplikasi android yang digunakan. Untuk membuat database, langkah pertama yang perlu dilakukan adalah membuka konsol Firebase (<https://console.firebaseio.google.com/>) pada web browser menggunakan akun Google yang akan digunakan, setelah itu membuat proyek baru, ataupun menggunakan proyek yang sudah ada.



Gambar 4.1 Konsol Firebase

Setelah proyek berhasil dibuat atau dipilih, buka proyek tersebut lalu pilih menu Firestore pada bagian kiri halaman proyek yang dibuka, lalu buat *database* baru jika *database* masih belum dibuat. Terakhir ubah peraturan pada database agar dapat di atur dari aplikasi Android.



Gambar 4.2 Peraturan *database*

#### 4.2 Implementasi Aplikasi Android yang digunakan

Aplikasi android yang digunakan ini menggunakan kode program yang dibuat dengan menggunakan bahasa pemrograman C#, dengan Visual Studio Code sebagai *Integrated Development Environment (IDE)*, dan digunakan untuk *script* pada *game engine* Unity. Alasan penggunaan bahasa pemrograman C# dalam pembuatan aplikasi ini adalah karena bahasa saat pembuatan aplikasi yang dapat digunakan pada *game engine* Unity hanyalah bahasa pemrograman C#.

##### 4.2.1 Implementasi Adegan Peta pada Aplikasi Android yang digunakan

Pada Tabel 4.1 merupakan tabel yang berisi daftar fungsi-fungsi yang digunakan dalam implementasi adegan peta beserta keterangan dan flowchart dari fungsi tersebut.

Tabel 4.1

Daftar Fungsi yang digunakan dalam Adegan Peta

Segmen Program	Nama Fungsi / Prosedur	Keterangan Fungsi	Gambar Flowchart
4.1	Properti Penanda	Menyimpan properti yang dibutuhkan pada suatu penanda	-
4.2	Inisialisasi Penanda	Inisialisasi properti pada penanda berdasarkan informasi <i>database</i>	-

4.3	Manajemen Daftar Penanda	Mengatur fungsi yang terdapat pada daftar penanda seperti pemilihan item dan pengubahan isi daftar penanda	Flowchart 3.3
4.4	Cek Lokasi	Mengecek apakah lokasi perangkat yang digunakan berada di dalam penanda yang dipilih oleh pengguna	Flowchart 3.4
4.5	Cek Gambar	Mengecek apakah gambar yang dilacak oleh perangkat yang digunakan adalah gambar yang sesuai	Flowchart 3.5
4.6	Tambah Penanda	Mengatur fungsi admin untuk menambah penanda	Flowchart 3.6
4.7	Edit Penanda	Mengatur fungsi admin untuk mengedit penanda	Flowchart 3.7
4.8	Hapus Penanda	Mengatur fungsi admin untuk menghapus penanda	Flowchart 3.8
4.9	Manajemen Fokus Kamera	Mengganti fokus pada kamera di adegan peta	-

#### 4.2.1.1 Implementasi Properti Penanda

Properti Penanda merupakan sebuah kode yang digunakan untuk menyimpan properti yang dibutuhkan pada suatu penanda, berupa nama penanda yang digunakan sebagai ID dan bersifat unik, lalu deskripsi penanda untuk memberi keterangan pada suatu penanda, ID karakter *default* penanda, dan lokasi dunia nyata dari penanda berupa garis lintang dan garis bujur. Segmen program ini dapat dilihat pada Segmen 4.1.

Segmen 4.1 Properti penanda

```
public class MarkerProperties : MonoBehaviour
{
    public string mName;
    public string mDesc;
    public string mChar;
    public double mLat;
    public double mLong;
}
```

#### 4.2.1.2 Implementasi Inisialisasi Penanda

Inisialisasi Penanda merupakan sebuah kode yang digunakan untuk menginisialisasi properti pada penanda berdasarkan informasi *database*, lalu setiap penanda yang ada pada *database* juga ditambahkan pada daftar penanda. Setelah itu, ketika semua penanda sudah selesai ditambahkan, maka akan dibuat daftar penanda pada UI aplikasi. Segmen program ini dapat dilihat pada Segmen 4.2.

Segmen 4.2 Inisialisasi penanda

```
// INITIALIZE MARKER
_spawnedObjects = new List<GameObject>();

// GET FROM DATABASE
db = FirebaseFirestore.DefaultInstance;
CollectionReference markersRef = db.Collection("markers");
markersRef.GetSnapshotAsync().ContinueWithOnMainThread(task =>
{
    QuerySnapshot snapshot = task.Result;
    foreach (DocumentSnapshot document in snapshot.Documents)
    {
        Dictionary<string, object> documentDictionary = document.ToDictionary();
        Vector2d tPos;
        tPos.x = double.Parse(documentDictionary["lat"].ToString());
        tPos.y = double.Parse(documentDictionary["long"].ToString());
        Vector3 tGamePos = _map.GeoToWorldPosition(tPos);
        GameObject tObj = Instantiate(markerObject, tGamePos, Quaternion.identity);

        tObj.GetComponent<MarkerProperties>().mName = document.Id;
        tObj.GetComponent<MarkerProperties>().mDesc = documentDictionary["desc"].ToString();
        tObj.GetComponent<MarkerProperties>().mChar = documentDictionary["char"].ToString();
        tObj.GetComponent<MarkerProperties>().mLat = tPos.x;
        tObj.GetComponent<MarkerProperties>().mLong = tPos.y;
        tObj.SetActive(false);
        _spawnedObjects.Add(tObj);

        listManager.addToList(document.Id, documentDictionary["desc"].ToString(),
_m_DropOptions[int.Parse(documentDictionary["char"].ToString())], tPos);
    }
    listManager.generateList();
});
```

#### 4.2.1.3 Implementasi Manajemen Daftar Penanda

Manajemen daftar Penanda merupakan sebuah kode yang digunakan untuk mengatur fungsi yang terdapat pada daftar penanda seperti pemilihan item dan pengubahan isi daftar penanda. Pengubahan isi daftar penanda berupa menambahkan item ke daftar penanda dan menghapus item dari daftar penanda, selain itu juga terdapat fungsi untuk membuat daftar penanda pada UI aplikasi, dan fungsi untuk mengecek apakah suatu nama terdapat pada daftar penanda atau tidak. Fungsi untuk mengecek nama ini digunakan untuk mengetahui apakah suatu nama penanda sudah digunakan atau belum, sehingga tidak ada dua penanda dengan nama yang sama. Segmen program ini dapat dilihat pada Segmen 4.3.

### Segmen 4.3 Manajemen daftar penanda

```
public class ListManager : MonoBehaviour
{
    [Serializable]
    public struct sMarker : IEquatable<sMarker>
    {
        public string Name;
        public string Description;
        public string Character;
        public Vector2d Position;

        public bool Equals(sMarker other)
        {
            return (this.Name.Equals(other.Name));
        }
    }
    List<sMarker> listMarker = new List<sMarker>();

    public GameObject listWindow;
    public GameObject buttonTemplate;
    public CameraController camController;
    public AbstractMap _map;

    void ItemClicked(int itemIndex)
    {
        listWindow.SetActive(false);
        camController.focusTo(_map.GeoToWorldPosition(listMarker[itemIndex].Position, true));
    }

    public void generateList()
    {
        // DELETE ALL
        foreach (Transform child in transform){
            if (child.name != "btnTemplate")
                GameObject.Destroy(child.gameObject);
        }

        for (int i = 0; i < listMarker.Count; i++)
        {
            GameObject g = Instantiate(buttonTemplate, transform);
            g.SetActive(true);
            g.transform.GetChild(0).GetComponent<TMPPro.TextMeshProUGUI>().text = listMarker[i].Name;
            g.transform.GetChild(1).GetComponent<TMPPro.TextMeshProUGUI>().text = listMarker[i].Description;
            g.transform.GetChild(2).GetComponent<TMPPro.TextMeshProUGUI>().text = "Character : " + listMarker[i].Character;
            g.GetComponent<Button>().AddEventListener(i, ItemClicked);
        }
    }

    public void addToList(string xName, string xDesc, string xChar, Vector2d xPos)
    {
        sMarker temp;
        temp.Name = xName;
        temp.Description = xDesc;
        temp.Character = xChar;
        temp.Position = xPos;
        listMarker.Add(temp);
    }

    public void deleteFromList(string xName)
    {
        listMarker.Remove(new sMarker { Name = xName });
    }

    public bool listContains(string xName)
    {
        if (listMarker.Contains(new sMarker { Name = xName }))
        {
            return true;
        }
        return false;
    }
}
```

#### 4.2.1.4 Implementasi Cek Lokasi

Cek lokasi merupakan sebuah kode yang digunakan untuk mengecek apakah lokasi perangkat yang digunakan berada di dalam penanda yang dipilih oleh pengguna dengan cara mengecek kolisi pada lokasi perangkat. Segmen program ini dapat dilihat pada Segmen 4.4.

##### Segmen 4.4 Cek Lokasi

```
void OnCollisionStay(Collision collision)
{
    if (!UIManager.getState() && !IsPointerOverUIObject() && Input.GetMouseButtonDown(0))
    {
        Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
        RaycastHit hit;
        if (Physics.Raycast(ray, out hit))
        {
            if (collision.transform.root.gameObject == hit.transform.root.gameObject)
            {
                GameObject adminManager = GameObject.Find("Admin Manager");
                if (adminManager != null)
                {
                    adminManager.GetComponent<AdminManager>().setDefaultChar(int.Parse
(hit.transform.root.GetComponent<MarkerProperties>().mChar));
                }
                sceneTransition.LoadScene(sceneName);
            }
        }
    }
}
```

#### 4.2.1.5 Implementasi Cek Gambar

Cek lokasi merupakan sebuah kode yang digunakan untuk mengecek apakah gambar yang dilacak oleh perangkat yang digunakan adalah gambar yang sesuai dengan cara mengecek gambar setiap kali ada perubahan gambar. Segmen program ini dapat dilihat pada Segmen 4.5.

##### Segmen 4.5 Cek gambar

```
ARTrackedImage logo;
void OnTrackedImagesChanged(ARTrackedImagesChangedEventArgs eventArgs)
{
    foreach (var trackedImage in eventArgs.added)
    {
        logo = trackedImage;
        buttonGO.SetActive(true);
        textInfo.text = "Press the GO button";
    }
    foreach (var trackedImage in eventArgs.updated)
    {
        if (trackedImage == logo)
        {
            buttonGO.SetActive(true);
            textInfo.text = "Press the GO button";
        }
    }
}
```

#### 4.2.1.6 Implementasi Tambah Penanda

Tambah penanda merupakan sebuah fungsi admin untuk menambah penanda. Fungsi ini diawali dengan pengecekan nama yang sudah digunakan, setelah itu membuat dan mengatur properti pada penanda, lalu menambah data informasi properti penanda ke daftar penanda dan *database*, dan terakhir menutup maupun menghapus data panel properti. Segmen program ini dapat dilihat pada Segmen 4.6.

Segmen 4.6 Tambah penanda

```
public void saveMarker()
{
    // CHECK NAME ALREADY EXIST
    if (listManager.listContains(propName.text))
    {
        showToast("Name already used", 1);
    }
    else
    {
        GameObject tObj = Instantiate(markerObject, tTarget, Quaternion.identity);
        Vector2d tPos = _map.WorldToGeoPosition(tTarget);
        tObj.GetComponent<MarkerProperties>().mName = propName.text;
        tObj.GetComponent<MarkerProperties>().mDesc = propDesc.text;
        tObj.GetComponent<MarkerProperties>().mChar = propChar.value.ToString();
        tObj.GetComponent<MarkerProperties>().mLat = tPos.x;
        tObj.GetComponent<MarkerProperties>().mLong = tPos.y;
        _spawnedObjects.Add(tObj);

        // SAVE TO LIST
        listManager.addToList(propName.text, propDesc.text, m_DropOptions[int.Parse(propChar.value.ToString())], tPos);
        listManager.generateList();

        // SAVE TO DATABASE
        DocumentReference docRef = db.Collection("markers").Document(propName.text);
        Dictionary<string, object> markerDict = new Dictionary<string, object>
        {
            { "lat", tPos.x },
            { "long", tPos.y },
            { "desc", propDesc.text },
            { "char", propChar.value.ToString() },
        };
        docRef.SetAsync(markerDict).ContinueWithOnMainThread(task =>
        {
            Debug.Log("Data Added");
        });
        closeProperties();
        propName.text = "";
        propDesc.text = "";
        propChar.value = 0;
    }
}
```

#### 4.2.1.7 Implementasi Edit Penanda

Edit penanda merupakan sebuah fungsi admin untuk mengedit penanda. Fungsi ini diawali dengan pengecekan nama yang sudah digunakan, setelah itu mengatur properti baru pada penanda, lalu mengubah data informasi properti penanda dari daftar penanda dan *database*, dan terakhir menutup maupun menghapus data panel edit. Segmen program ini dapat dilihat pada Segmen 4.7.

Segmen 4.7 Edit penanda

```
public void saveEdit()
{
    // CHECK NAME ALREADY EXIST
    if (editName.text != oldName && listManager.listContains(editName.text))
    {
        showToast("Name already used", 1);
    }
    else
    {
        // SET NEW PROPERTIES
        MarkerProperties temp = editObject.GetComponent<MarkerProperties>();
        temp.mName = editName.text;
        temp.mDesc = editDesc.text;
        temp.mChar = editChar.value.ToString();

        // MAKE TEMP POS
        Vector2d tPos = new Vector2d(editObject.GetComponent<MarkerProperties>().mLat,
editObject.GetComponent<MarkerProperties>().mLong);

        // EDIT FROM LIST
        listManager.deleteFromList(oldName);
        listManager.addToList(temp.mName, temp.mDesc, m_DropOptions[int.Parse
(editChar.value.ToString())], tPos);
        listManager.generateList();

        // EDIT FROM DATABASE
        DocumentReference markerRef = db.Collection("markers").Document(oldName);
        markerRef.DeleteAsync().ContinueWithOnMainThread(task =>
{
            DocumentReference docRef = db.Collection("markers").Document(temp.mName);
            Dictionary<string, object> markerDict = new Dictionary<string, object>
            {
                { "lat", temp.mLat },
                { "long", temp.mLong },
                { "desc", temp.mDesc },
                { "char", temp.mChar },
            };
            docRef.SetAsync(markerDict);
        });

        closeEdit();
        editName.text = "";
        editDesc.text = "";
        editChar.value = 0;
    }
}
```

#### 4.2.1.8 Implementasi Hapus Penanda

Hapus penanda merupakan sebuah fungsi admin untuk menghapus penanda. Fungsi ini diawali dengan membuat variabel privat untuk menyimpan nama penanda yang akan dihapus, lalu ketika ada penanda yang mau dihapus, akan dicek apakah nama penanda sama dengan variabel privat. Apabila penanda pertama kali dipilih maka nama penanda akan berbeda dengan variabel privat, lalu akan dimunculkan notifikasi berupa *toast* untuk memilih penanda yang sama lagi sebagai konfirmasi untuk penghapusan, dan variabel privat diset menjadi nama penanda yang dipilih. Setelah itu jika penanda yang sama dipilih lagi, maka variabel privat akan sama dengan penanda yang dipilih, setelah itu baru penanda dihapus dari peta, daftar penanda, dan *database*. Segmen program ini dapat dilihat pada Segmen 4.8.

Segmen 4.8 Hapus penanda

```
private string delName = "";
void Update()
{
    if (!IsPointerOverUIObject() && Input.GetMouseButtonDown(0))
    {
        Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);
        RaycastHit hit;
        if (Physics.Raycast(ray, out hit))
        {
            GameObject temp = hit.transform.root.gameObject;
            if (temp.tag == "Marker")
            {
                if (delName == temp.GetComponent<MarkerProperties>().mName)
                {
                    _spawnedObjects.Remove(temp);
                    temp.Destroy();

                    delState = false;

                    // DELETE FROM LIST
                    listManager.deleteFromList(delName);
                    listManager.generateList();

                    // DELETE FROM DATABASE
                    DocumentReference markerRef = db.Collection("markers").Document(delName);
                    markerRef.DeleteAsync();
                }
                else
                {
                    showToast("Select Again to Delete", 1);
                    delName = temp.GetComponent<MarkerProperties>().mName;
                }
            }
        }
    }
}
```

#### 4.2.1.9 Implementasi Manajemen Kamera

Manajemen kamera merupakan sebuah fungsi untuk mengganti fokus pada kamera di adegan peta. Fungsi untuk mengganti fokus terdapat 2 macam, yaitu fungsi untuk mengganti fokus ke suatu posisi, dan fungsi untuk mengganti fokus ke posisi pengguna pada adegan peta. Segmen program ini dapat dilihat pada Segmen 4.9.

##### Segmen 4.9 Manajemen kamera

```
public class CameraController : MonoBehaviour
{
    [SerializeField]
    Camera cam;
    Vector3 previousPos = Vector3.zero;
    Vector3 deltaPos = Vector3.zero;

    void CamControl()
    {
        deltaPos = transform.position - previousPos;
        deltaPos.y = 0;
        cam.transform.position = Vector3.Lerp(cam.transform.position,
        cam.transform.position + deltaPos, Time.time);
        previousPos = transform.position;
    }

    public void focusTo(Vector3 x)
    {
        turnOffPlayerFocus();
        StopCoroutine(focusTransition(x));
        StartCoroutine(focusTransition(x));
    }

    IEnumerator focusTransition(Vector3 endPos, bool toPlayer = false)
    {
        Vector3 startPos = cam.transform.position;
        endPos.y = startPos.y;
        endPos.z -= startPos.y;
        float counter = 0f;
        float duration = 0.5f;
        while (counter < duration)
        {
            counter += Time.deltaTime;
            if (toPlayer)
            {
                endPos = transform.position;
                endPos.y = startPos.y;
                endPos.z -= startPos.y;
            }
            float xCurrent = Mathf.Lerp(startPos.x, endPos.x, counter / duration);
            float zCurrent = Mathf.Lerp(startPos.z, endPos.z, counter / duration);
            cam.transform.position = new Vector3(xCurrent, startPos.y, zCurrent);
            yield return null;
        }
    }
}
```

#### Segmen 4.9 Manajemen kamera (sambungan)

```
public void refocusPlayer()
{
    turnOnPlayerFocus();
    StopCoroutine(focusTransition(transform.position, true));
    StartCoroutine(focusTransition(transform.position, true));
}

public GameObject refocusButton;

public void turnOffPlayerFocus()
{
    isPlayerFocused = false;
    refocusButton.SetActive(true);
}
public void turnOnPlayerFocus()
{
    isPlayerFocused = true;
    refocusButton.SetActive(false);
}

public bool isPlayerFocused = true;

void Update()
{
    if (isPlayerFocused)
    {
        if (cam.GetComponent<Mapbox.Examples.CameraMovement>()._shouldDrag &&
!IsPointerOverUIObject() && Input.GetMouseButton(0))
            turnOffPlayerFocus();
        else
            CamControl();
    }
}
```

#### 4.2.2 Implementasi Adegan AR pada Aplikasi Android yang digunakan

Pada Tabel 4.2 merupakan tabel yang berisi daftar fungsi-fungsi yang digunakan dalam implementasi adegan AR beserta keterangan dan flowchart dari fungsi tersebut.

Tabel 4.2

Daftar Fungsi yang digunakan dalam Adegan AR

Segment Program	Nama Fungsi / Prosedur	Keterangan Fungsi	Gambar Flowchart
4.10	Tampilkan dan Ganti Posisi	Menampilkan objek 3D virtual yang belum ditampilkan dan menentukan posisinya, atau memindah posisi objek yang sudah ditampilkan	Flowchart 3.9

4.11	Ganti Animasi	Mengganti animasi karakter yang sudah disediakan	Flowchart 3.10
4.12	Ganti Rotasi	Mengganti rotasi objek yang sudah ditampilkan, pada sumbu Y ( <i>yaw</i> )	Flowchart 3.11
4.13	Ganti Skala	Mengganti skala objek yang sudah ditampilkan, pada ketiga sumbu	Flowchart 3.11
4.14	Ganti Model	Mengganti model objek menjadi karakter lain atau model hasil <i>import</i>	Flowchart 3.12
4.15	<i>Import</i> Model	Membuat model dari <i>file</i> dengan ekstensi OBJ pada penyimpanan internal perangkat	Flowchart 3.12
4.16	Ambil Foto	Mengambil gambar dari kamera dan objek 3D	Flowchart 3.13
4.17	<i>Preview</i> Foto	Menyimpan atau menghapus hasil gambar yang sudah diambil	Flowchart 3.13
4.18	Reset Sesi AR	Mengulang pelacakan pada adegan AR dari awal dan menghancurkan objek yang sudah ditampilkan	Flowchart 3.14
4.19	Mode Debug	Menyalakan atau mematikan penghitung <i>Frames Per Second</i> , penanda arah cahaya, dan <i>plane visualizer</i> yang selalu aktif dan juga melakukan reset sesi AR	Flowchart 3.15
4.20	Karakter Debug	Menyalakan atau mematikan karakter dan bidang permukaan yang selalu aktif	Flowchart 3.15
4.21	Estimasi Cahaya	Menyalakan atau mematikan penggunaan metode estimasi cahaya	Flowchart 3.15
4.22	Shader Penerima Bayangan	Shader yang digunakan pada material visualisasi pelacakan bidang permukaan	-

#### 4.2.2.1 Implementasi Tampilkan dan Ganti Posisi

Tampilkan dan ganti posisi merupakan sebuah fungsi untuk menampilkan objek 3D virtual yang belum ditampilkan dan menentukan posisinya, atau memindah posisi objek yang

sudah ditampilkan. Pengecekan untuk menjalankan menampilkan atau mengganti posisi berada pada bagian fungsi “Update()” yang dijalankan setiap *frame*. Pengecekannya berupa apakah posisi yang disentuh tidak berada di atas objek UI, dan apakah posisi yang disentuh berada pada bidang permukaan yang sudah dilacak, jika keduanya benar maka akan dicek apakah objek sudah ditampilkan atau belum. Jika objek belum ditampilkan maka objek akan ditampilkan, lalu dirotasi awal pada sumbu Y (*yaw*) agar model dapat mengarah ke pengguna, menyetel pengatur animasi (animator), dan mematikan visualisasi pelacakan bidang permukaan, jika objek sudah ditampilkan maka objek hanya dipindah posisinya. Segmen program ini dapat dilihat pada Segmen 4.10.

#### Segmen 4.10 Tampilkan dan ganti posisi

```
public float yAxisRotateValue = 180;
public ARUIManager uiManager;

void Update()
{
    if (!IsPointOverUIObject(touchPosition) && m_RaycastManager.Raycast(touchPosition,
s_Hits, TrackableType.PlaneWithinPolygon))
    {
        var hitPose = s_Hits[0].pose;

        if (spawnedObject == null)
        {
            spawnedObject = Instantiate(m_PlacedPrefab, hitPose.position,
hitPose.rotation);
            spawnedObject.SetActive(true);
            spawnedObject.transform.Rotate(0.0f, yAxisRotateValue, 0.0f, Space.World);
            spawnedObject.transform.localScale = new Vector3(defaultObjectScale,
defaultObjectScale, defaultObjectScale);

            // SET ANIM & PLANE
            setAnim();
            uiManager.turnHintOn(false);
        }
        else
        {
            spawnedObject.transform.position = hitPose.position;
        }
    }
}
```

#### 4.2.2.2 Implementasi Ganti Animasi

Tampilkan dan ganti posisi merupakan sebuah fungsi untuk mengganti animasi karakter yang sudah disediakan. Tombol yang disediakan pada aplikasi dapat menjalankan fungsi “animNext()” atau fungsi “animBack()”, lalu setelah tombol tersebut dipencet, maka akan disetel variabel yang sesuai pada animator untuk mengetahui tombol mana yang sudah dipencet. Pengecekan untuk menjalankan ganti animasi berada pada bagian fungsi “Update()” yang dijalankan setiap *frame*. Pengecekannya berupa apakah objek sudah ditampilkan, dan apakah

objek tersebut memiliki animator, jika keduanya benar maka animasi karakter akan diubah sesuai dengan tombol yang dipilih Segmen program ini dapat dilihat pada Segmen 4.11.

#### Segmen 4.11 Ganti animasi

```
public void animNext()
{
    if (spawnedObject != null && anim != null)
        anim.SetBool("Next", true);
}

public void animBack()
{
    if (spawnedObject != null && anim != null)
        anim.SetBool("Back", true);
}

void Update()
{
    if (spawnedObject != null && anim != null)
    {
        if (anim.GetBool("Next"))
        {
            currentState = anim.GetCurrentAnimatorStateInfo(0);
            if (previousState.fullPathHash != currentState.fullPathHash)
            {
                anim.SetBool("Next", false);
                previousState = currentState;
            }
        }

        if (anim.GetBool("Back"))
        {
            currentState = anim.GetCurrentAnimatorStateInfo(0);
            if (previousState.fullPathHash != currentState.fullPathHash)
            {
                anim.SetBool("Back", false);
                previousState = currentState;
            }
        }
    }
}
```

#### 4.2.2.3 Implementasi Ganti Rotasi

Ganti rotasi merupakan sebuah fungsi untuk mengganti rotasi objek yang sudah ditampilkan, pada sumbu Y (*yaw*). Setiap kali fungsi untuk mengganti rotasi dipanggil, maka rotasi objek akan diganti sebesar 15 derajat pada sumbu Y global. Segmen program ini dapat dilihat pada Segmen 4.12.

#### Segmen 4.12 Ganti rotasi

```
public void rotateClockwise()
{
    if (spawnedObject != null)
        spawnedObject.transform.Rotate(0.0f, 15, 0.0f, Space.World);
}

public void rotateCounterClockwise()
{
    if (spawnedObject != null)
        spawnedObject.transform.Rotate(0.0f, -15, 0.0f, Space.World);
}
```

#### 4.2.2.4 Implementasi Ganti Skala

Ganti skala merupakan sebuah fungsi untuk mengganti skala objek yang sudah ditampilkan, pada ketiga sumbu. Setiap kali fungsi untuk mengganti skala dipanggil, maka skala objek akan diganti sebesar 0.1 pada ketiga sumbu dari skala awalnya. Segmen program ini dapat dilihat pada Segmen 4.13.

#### Segmen 4.13 Ganti skala

```
public void scaleUp()
{
    if (spawnedObject != null)
    {
        Vector3 temp = spawnedObject.transform.localScale;
        temp.x += 0.1f;
        temp.y += 0.1f;
        temp.z += 0.1f;
        spawnedObject.transform.localScale = temp;
    }
}

public void scaleDown()
{
    if (spawnedObject != null)
    {
        Vector3 temp = spawnedObject.transform.localScale;
        temp.x -= 0.1f;
        temp.y -= 0.1f;
        temp.z -= 0.1f;
        spawnedObject.transform.localScale = temp;
    }
}
```

#### 4.2.2.5 Implementasi Ganti Model

Ganti model merupakan sebuah fungsi untuk mengganti model objek menjadi karakter lain atau model hasil *import*. Setiap kali fungsi untuk mengganti model dipanggil, maka model objek akan diganti pada suatu siklus karakter yang sudah disediakan, dan jika ada model hasil *import* maka model tersebut diletakkan pada paling belakang siklus. Segmen program ini dapat dilihat pada Segmen 4.14.

#### Segmen 4.14 Ganti model

```
public void charNext()
{
    GameObject loadedObj = uiManager.getLoadedObj();
    if (currentModel >= characterModels.Length - 1 && loadedObj == null || currentModel >=
characterModels.Length && loadedObj != null)
        currentModel = -1;
    currentModel++;
    updateChar();
}

public void charBack()
{
    GameObject loadedObj = uiManager.getLoadedObj();
    if (currentModel <= 0 && loadedObj == null)
        currentModel = characterModels.Length;
    else if (currentModel <= 0 && loadedObj != null)
        currentModel = characterModels.Length + 1;
    currentModel--;
    updateChar();
}

public void updateChar()
{
    GameObject loadedObj = uiManager.getLoadedObj();
    if (currentModel >= characterModels.Length)
        m_PlacedPrefab = loadedObj;
    else
        m_PlacedPrefab = characterModels[currentModel];

    if (spawnedObject != null)
    {
        Transform temp = spawnedObject.transform;
        spawnedObject.Destroy();
        if (currentModel >= characterModels.Length)
            spawnedObject = Instantiate(loadedObj, temp.position, temp.rotation);
        else
            spawnedObject = Instantiate(characterModels[currentModel], temp.position, temp.rotation);
        spawnedObject.SetActive(true);
        spawnedObject.transform.localScale = new Vector3(defaultObjectScale, defaultObjectScale,
defaultObjectScale);
        setAnim();
    }
}

private void setAnim()
{
    anim = null;
    anim = spawnedObject.GetComponent<Animator>();
    if (anim != null)
    {
        currentState = anim.GetCurrentAnimatorStateInfo(0);
        previousState = currentState;
    }
}
```

#### 4.2.2.6 Implementasi Import Model

*Import* model merupakan sebuah fungsi untuk membuat model dari *file* dengan ekstensi OBJ pada penyimpanan internal perangkat. Apabila *file* yang berada pada penyimpanan eksternal dipilih, maka tidak akan diimpor. Segmen program ini dapat dilihat pada Segmen 4.15.

Segmen 4.15 *Import* model

```
// IMPORT OBJ
string filePath = FileBrowser.Result[0];
if (Application.platform != RuntimePlatform.Android || filePath.Contains("/storage/emulated/0"))
{
    if (loadedObj != null)
    {
        listCharName.RemoveAt(characterModels.Length);
        loadedObj.Destroy();
    }
    loadedObj = new OBJLoader().Load(filePath);
    loadedObj.transform.localScale = new Vector3(1, 1, 1);
    loadedObj.SetActive(false);

    // SET NORMAL CHAR
    scriptPOP.updateObj();

    // SET DEBUG CHAR
    Transform temp = character.transform;
    if (character.activeSelf)
    {
        character.Destroy();
        character = Instantiate(loadedObj, temp.position, temp.rotation);
        character.SetActive(true);
    }
    else
    {
        character.Destroy();
        character = Instantiate(loadedObj, temp.position, temp.rotation);
        character.SetActive(false);
    }
    currentModel = characterModels.Length;
    listCharName.Add(loadedObj.name);
    updateTextChar();
}
else
{
    showToast("Please choose files from internal storage", 1);
}
```

#### 4.2.2.7 Implementasi Ambil Foto

Ambil foto merupakan sebuah fungsi untuk mengambil gambar dari kamera dan objek 3D. Setelah foto diambil, maka foto akan ditampilkan pada panel *preview*. Segmen program ini dapat dilihat pada Segmen 4.16.

Segmen 4.16 Ambil foto

```
public void ShotNow()
{
    StopCoroutine(SSCapture());
    StartCoroutine(SSCapture());
}
```

#### Segmen 4.16 Ambil foto (sambungan)

```
IEnumerator SSCapture()
{
    myFileName = fileNamePrefix + System.DateTime.Now.ToString(fileTimeStampFormat) + fileFormat;
    tempFileName = Application.persistentDataPath + "/" + myFileName;

    UI.SetActive(!UI.activeSelf);
    pnlFlash.SetActive(!pnlFlash.activeSelf);
    yield return new WaitForSeconds(flashDuration);
    pnlFlash.SetActive(!pnlFlash.activeSelf);

    ScreenCapture.CaptureScreenshot(myFileName);
    yield return new WaitForSeconds();
    canvasPreview.SetActive(!canvasPreview.activeSelf);

    Texture2D texture = null;
    byte[] fileBytes;

    while (true)
    {
        if (File.Exists(tempFileName))
        {
            fileBytes = File.ReadAllBytes(tempFileName);
            texture = new Texture2D(2, 2, TextureFormat.RGB24, false);
            texture.LoadImage(fileBytes);

            Sprite sp = Sprite.Create(texture, new Rect(0, 0, texture.width, texture.height),
            new Vector2(0.5f, 0.5f));
            pnlPreview.GetComponent<Image>().sprite = sp;

            isAndroid = true;
            break;
        }
        else if (File.Exists(myFileName))
        {
            fileBytes = File.ReadAllBytes(myFileName);
            texture = new Texture2D(2, 2, TextureFormat.RGB24, false);
            texture.LoadImage(fileBytes);

            Sprite sp = Sprite.Create(texture, new Rect(0, 0, texture.width, texture.height),
            new Vector2(0.5f, 0.5f));
            pnlPreview.GetComponent<Image>().sprite = sp;

            isAndroid = false;
            break;
        }
        else
        {
            Debug.Log("File doesn't exist yet");
        }
        yield return new WaitForSeconds(0.05f);
    }
}
```

#### 4.2.2.8 Implementasi *Preview* Foto

*Preview* foto merupakan sebuah fungsi untuk menyimpan dan menghapus hasil gambar yang sudah diambil. Jika aplikasi dijalankan pada sistem operasi android, maka penyimpanan gambar dilakukan dengan menggunakan *plugin “Unity Native Gallery”* yang berada pada halaman web <https://github.com/yasirkula/UnityNativeGallery>. Segmen program ini dapat dilihat pada Segmen 4.17.

#### Segmen 4.17 Preview foto

```
public void deleteSS()
{
    if (isAndroid)
        File.Delete(tempFileName);
    else
        File.Delete(myFileName);
    canvasPreview.SetActive(!canvasPreview.activeSelf);
    UI.SetActive(!UI.activeSelf);
}

public void saveSS()
{
    if (isAndroid)
    {
        NativeGallery.Permission permission = NativeGallery.SaveImageToGallery
(tempFileName, fileFolder, myFileName, (success, path) => Debug.Log("Media save
result: " + success + " " + path));
        Debug.Log("Permission result: " + permission);
        deleteSS();
    }
    else
    {
        canvasPreview.SetActive(!canvasPreview.activeSelf);
        UI.SetActive(!UI.activeSelf);
    }
}
```

#### 4.2.2.9 Implementasi Reset Sesi AR

Reset sesi AR merupakan sebuah fungsi untuk mengulang pelacakan pada adegan AR dari awal dan menghancurkan objek yang sudah ditampilkan. Segmen program ini dapat dilihat pada Segmen 4.18.

#### Segmen 4.18 reset sesi AR

```
public void Reset()
{
    if (subsystem != null)
        subsystem.Reset();
    if (state > ARSessionState.Ready)
        state = ARSessionState.SessionInitializing;
}
public void destroySpawnedObject()
{
    if (spawnedObject != null)
    {
        spawnedObject.Destroy();
        spawnedObject = null;
        uiManager.turnHintOn(true);
    }
}
```

#### 4.2.2.10 Implementasi Mode Debug

Mode debug merupakan sebuah fungsi untuk menyalakan atau mematikan penghitung *Frames Per Second*, penanda arah cahaya, dan *plane visualizer* yang selalu aktif, selain itu juga melakukan reset sesi AR. Segmen program ini dapat dilihat pada Segmen 4.19.

Segmen 4.19 Mode debug

```
public void toggleDebug()
{
    txtFPS.gameObject.SetActive(!txtFPS.gameObject.activeSelf);
    dLight.GetComponent<UnityEngine.XR.ARFoundation.Samples.HDRLightEstimation>().toggleUseArrow();
    if (transparentState)
        transparentState = false;
    else
        transparentState = true;
}

void Update()
{
    if (txtFPS.gameObject.activeSelf && Time.timeScale == 1)
    {
        float fps = 1 / Time.unscaledDeltaTime;
        txtFPS.text = $"FPS: {Mathf.Ceil(fps)}";
        if (fps > maxFrameRate)
            maxFrameRate = (int)fps;
        if (fps < minFrameRate)
            minFrameRate = (int)fps;
        frameSum += fps;
        frameCounted++;
        txtFPS.text += "\nAVG: " + (frameSum / frameCounted).ToString("f2");
        txtFPS.text += "\nMAX: " + maxFrameRate;
        txtFPS.text += "\nMIN: " + minFrameRate;
    }
}

public bool UseArrow = true;
public void toggleUseArrow()
{
    if (UseArrow)
        UseArrow = false;
    else
        UseArrow = true;
}

void FrameChanged(ARCameraFrameEventArgs args)
{
    if (args.lightEstimation.mainLightDirection.HasValue)
    {
        if (UseArrow && arrow)
        {
            arrow.gameObject.SetActive(true);
            arrow.rotation = Quaternion.LookRotation(mainLightDirection.Value);
        }
    }
}
```

#### 4.2.2.11 Implementasi Karakter Debug

Karakter debug merupakan sebuah fungsi untuk menyalakan atau mematikan karakter dan bidang permukaan yang selalu aktif. Segmen program ini dapat dilihat pada Segmen 4.20.

Segmen 4.20 Karakter debug

```
public void toggleChar()
{
    debugPlane.SetActive(!debugPlane.activeSelf);
    character.SetActive(!character.activeSelf);
}
```

#### 4.2.2.12 Implementasi Estimasi Cahaya

Estimasi cahaya merupakan sebuah fungsi untuk menyalakan atau mematikan penggunaan metode estimasi cahaya, lalu melakukan reset sesi AR. Pada saat adegan AR dibuka, *state* metode estimasi cahaya disimpan sehingga dapat digunakan ketika ingin dinyalakan lagi. Segmen program ini dapat dilihat pada Segmen 4.21.

Segmen 4.21 Estimasi cahaya

```
private UnityEngine.XR.ARFoundation.LightEstimation leState;
void Start()
{
    leState = arSessionOrigin.transform.GetChild
(0).GetComponent<UnityEngine.XR.ARFoundation.ARCameraManager>().requestedLightEstimation;
}

public void toggleLE()
{
    UnityEngine.XR.ARFoundation.ARCameraManager temp = arSessionOrigin.transform.GetChild
(0).GetComponent<UnityEngine.XR.ARFoundation.ARCameraManager>();
    if (temp.requestedLightEstimation == leState)
    {
        temp.requestedLightEstimation = UnityEngine.XR.ARFoundation.LightEstimation.None;
        dLight.transform.rotation = Quaternion.Euler(90, 0, 0);
        dLight.GetComponent<Light>().intensity = 1;
    }
    else
        temp.requestedLightEstimation = leState;
}
```

#### 4.2.2.13 Implementasi *Shader* Penerima Bayangan

*Shader* penerima bayangan merupakan *shader* yang digunakan pada material visualisasi pelacakan bidang permukaan. Segmen program ini dapat dilihat pada Segmen 4.22.

Segmen 4.22 *Shader* penerima bayangan

```
Shader "URP AR Shadow Receiver"
{
    Properties
    {
        _ShadowColor("Shadow Color", Color) = (0.35,0.4,0.45,1.0)
    }

    SubShader
    {
        Tags
        {
            "RenderPipeline" = "UniversalPipeline"
            "RenderType" = "Transparent"
            "Queue" = "Transparent-1"
        }
    }
}
```

Segmen 4.22 *Shader* penerima bayangan (sambungan)

```

Pass
{
    Name "ForwardLit"
    Tags { "LightMode" = "UniversalForward" }

    Blend DstColor Zero, Zero One
    Cull Back
    ZTest LEqual
    ZWrite Off

    HLSLPROGRAM
    #pragma vertex vert
    #pragma fragment frag
    #pragma prefer_hlslcc gles
    #pragma exclude_renderers d3d11_9x
    #pragma target 2.0
    #pragma multi_compile _ _MAIN_LIGHT_SHADOWS
    #pragma multi_compile _ _MAIN_LIGHT_SHADOWS_CASCADE
    #pragma multi_compile _ _SHADOWS_SOFT
    #pragma multi_compile_fog
    #include "Packages/com.unity.render-pipelines.universal/ShaderLibrary/Lighting.hlsl"

    CBUFFER_START(UnityPerMaterial)
    float4 _ShadowColor;
    CBUFFER_END

    struct Attributes
    {
        float4 positionOS : POSITION;
        UNITY_VERTEX_INPUT_INSTANCE_ID
    };

    struct Varyings
    {
        float4 positionCS : SV_POSITION;
        float3 positionWS : TEXCOORD0;
        float fogCoord : TEXCOORD1;
        UNITY_VERTEX_INPUT_INSTANCE_ID
        UNITY_VERTEX_OUTPUT_STEREO
    };
    Varyings vert(Attributes input)
    {
        Varyings output = (Varyings)0;
        UNITY_SETUP_INSTANCE_ID(input);
        UNITY_TRANSFER_INSTANCE_ID(input, output);
        UNITY_INITIALIZE_VERTEX_OUTPUT_STEREO(output);
        VertexPositionInputs vertexInput = GetVertexPositionInputs(input.positionOS.xyz);
        output.positionCS = vertexInput.positionCS;
        output.positionWS = vertexInput.positionWS;
        output.fogCoord = ComputeFogFactor(vertexInput.positionCS.z);
        return output;
    }

    half4 frag(Varyings input) : SV_Target
    {
        UNITY_SETUP_INSTANCE_ID(input);
        UNITY_SETUP_STEREO_EYE_INDEX_POST_VERTEX(input);
        half4 color = half4(1,1,1,1);
#ifdef _MAIN_LIGHT_SHADOWS
        VertexPositionInputs vertexInput = (VertexPositionInputs)0;
        vertexInput.positionWS = input.positionWS;

        float4 shadowCoord = GetShadowCoord(vertexInput);
        half shadowAttenuation = MainLightRealtimeShadow(shadowCoord);
        color = lerp(half4(1,1,1,1), _ShadowColor, (1.0 - shadowAttenuation) * _ShadowColor.a);
        color.rgb = MixFogColor(color.rgb, half3(1,1,1), input.fogCoord);
#endif
        return color;
    }
    ENDHLSL
}
}

```

## 5. PENGUJIAN SISTEM

Pada bab ini akan dibahas mengenai hasil pengujian aplikasi yang telah dibuat beserta dengan pengoperasian aplikasinya. Pengujian sistem yang dilakukan meliputi empat hal utama, yaitu pengujian aplikasi secara keseluruhan, pengujian arah bayangan pada metode estimasi cahaya, pengujian perbedaan penggunaan *resource* pada metode estimasi cahaya, dan terakhir pengujian akurasi pelacakan lokasi pada sensor lokasi. Pengujian sistem dilakukan pada dua perangkat dengan spesifikasi yang berbeda. Perangkat pertama memiliki spesifikasi :

- Nama : Samsung Galaxy A8 (2018)
- Waktu rilis : 2018, Januari
- Sistem operasi : Android 9 Pie
- Prosesor : Exynos 7885 Octa-core (2x2.2 GHz & 6x1.6 GHz)
- GPU : Mali-G71
- RAM : 4GB

Perangkat kedua memiliki spesifikasi :

- Nama : Xiaomi Redmi Note 8 Pro
- Waktu rilis : 2019, 24 September
- Sistem operasi : Android 10
- Prosesor : Mediatek Helio G90T Octa-core (2x2.05 GHz & 6x2.0 GHz)
- GPU : Mali-G76 MC4
- RAM : 6GB

### 5.1 Pengujian Aplikasi

Pengujian aplikasi dilakukan pada keseluruhan fitur yang berada pada aplikasi dari pertama kali aplikasi dibuka, yaitu menu utama, sampai dengan penggunaan adegan AR untuk pengambilan foto bersama dengan karakter 3D virtual. Tujuannya adalah untuk mengetahui apakah fitur-fitur yang terdapat pada aplikasi dapat berjalan dengan baik. Pengujian aplikasi dilakukan menggunakan perangkat pertama (Samsung Galaxy A8) dengan spesifikasi yang tertulis pada bagian awal bab 5, pada lokasi di Universitas Kristen Petra.

#### 5.1.1 Pengujian Lokasi pada Universitas Kristen Petra

Pengujian lokasi dilakukan pada 6 Lokasi di UK Petra, yaitu Kolam Gedung P, Kolam Jodoh, Patung Torso, Kolam Gedung Q, Depan Gedung Q (Petranesian), dan Tangga Gedung Q.



Gambar 5.1 Pengujian pada kolam gedung P, kolam jodoh, dan patung torso (cahaya dari depan objek)

Dari gambar hasil pengujian di atas, dapat dilihat bahwa aplikasi dapat digunakan untuk mendapatkan output berupa foto bersama dengan karakter 3D virtual, selanjutnya, akan diuji apakah foto dapat diambil dari berbagai angle.



Gambar 5.2 Pengujian berbagai *angle* pada patung torso (cahaya dari depan torso)



Gambar 5.3 Pengujian berbagai *angle* pada patung torso dengan karakter lain (cahaya dari depan torso)

Selanjutnya akan dilakukan pengujian pengambilan foto dari berbagai *angle* pada lokasi lain di UK Petra (kolam gedung Q, depan gedung Q, dan tangga gedung Q) dengan karakter *default* masing-masing.



Gambar 5.4 Pengujian berbagai *angle* pada kolam gedung Q (Cahaya dari depan objek)



Gambar 5.5 Pengujian berbagai *angle* pada depan gedung Q (petranesian) (cahaya dari atas objek)

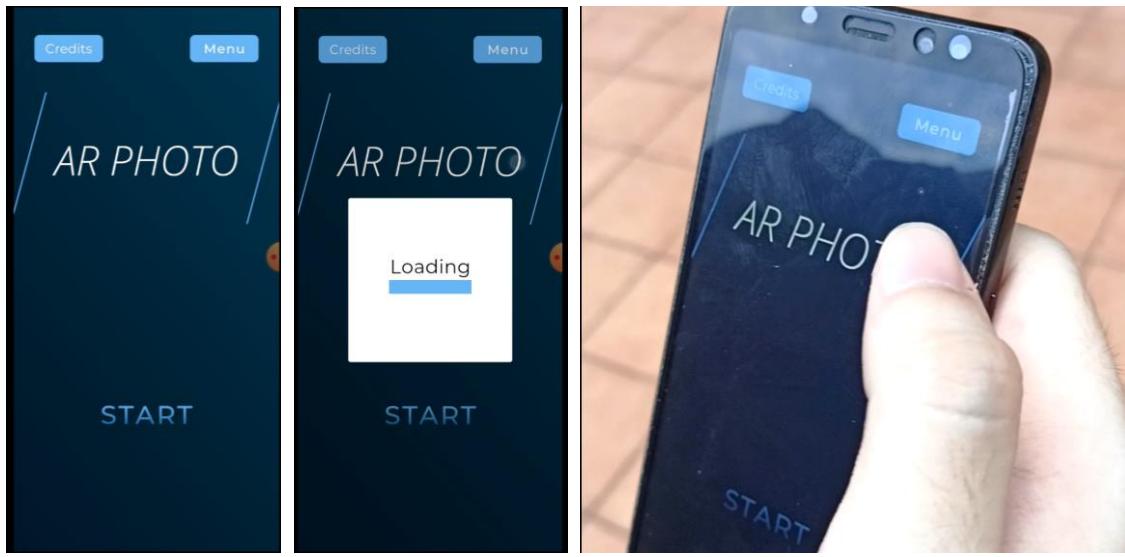


Gambar 5.6 Pengujian berbagai *angle* pada tangga gedung Q (cahaya dari bawah tangga / luar)

Dari gambar hasil pengujian di atas, dapat dilihat bahwa aplikasi dapat digunakan untuk mengambil foto dari berbagai *angle*, pada berbagai lokasi yang sudah ditentukan oleh admin, dengan berbagai karakter *default* ataupun pilihan.

### 5.1.2 Pengujian Menu Utama

Pada menu utama, pengguna dapat memilih untuk melanjutkan aplikasi dengan menekan bagian start pada layar. Selain itu, pengguna juga dapat melihat kredit, dan membuka menu dengan menekan tombolnya masing-masing, menu juga dapat dibuka ketika tombol kembali android ditekan. Pada menu terdapat pengaturan mode admin (untuk aplikasi versi admin), dan tombol untuk keluar dari aplikasi. Jika tombol untuk keluar dari aplikasi ditekan maka panel menu akan digantikan dengan panel konfirmasi, jika pengguna memilih untuk batal maka panel konfirmasi akan dikembalikan menjadi panel menu, dan jika pengguna memilih untuk keluar maka aplikasi akan ditutup.

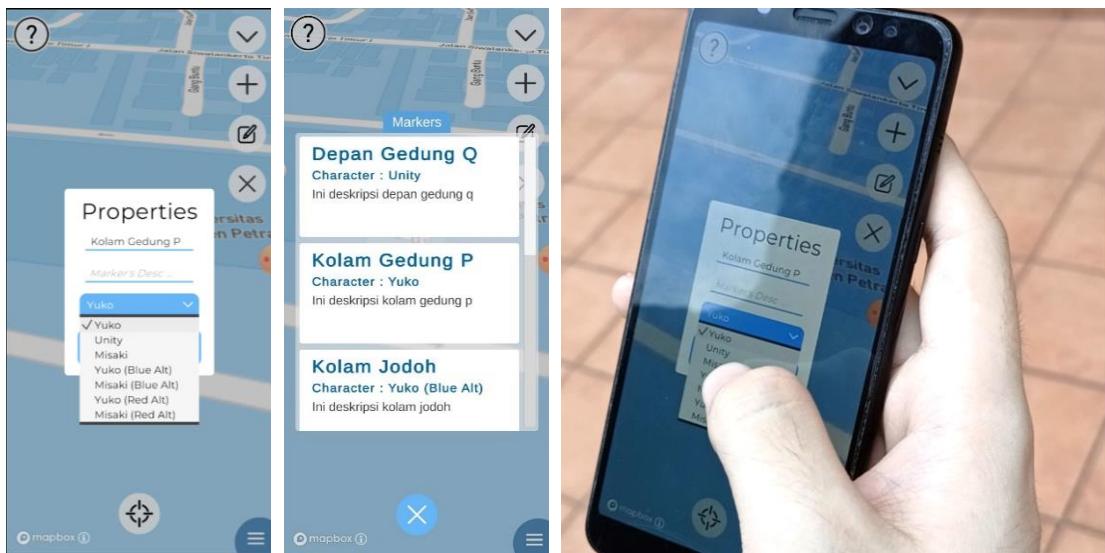


Gambar 5.7 Pengujian menu utama

Jika pengguna memilih untuk melanjutkan aplikasi maka akan dikeluarkan panel *loading bar* untuk menunjukkan statusnya. Setelah *loading bar* sudah penuh (sampai paling kanan) maka adegan menu utama ditutup dan digantikan dengan adegan map.

### 5.1.3 Pengujian Adegan Peta

Pada adegan peta, pengguna dapat melihat penjelasan adegan peta, memilih penanda, dan membuka menu. Pada menu terdapat tiga tombol untuk membuka daftar penanda, untuk berganti ke adegan *image tracking*, dan untuk kembali ke menu utama.



Gambar 5.8 Pengujian adegan peta

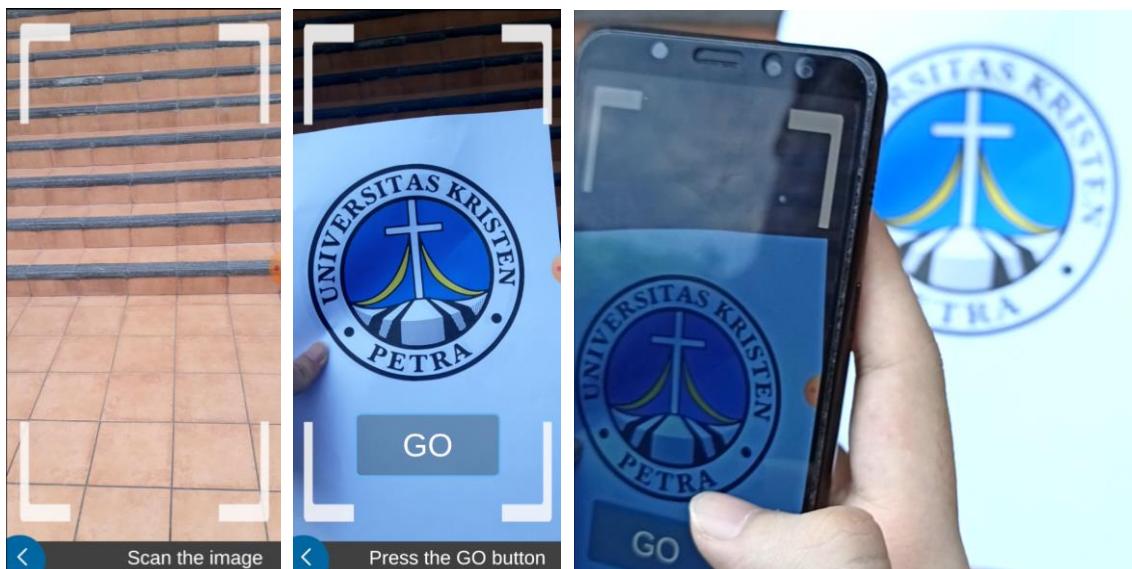
Daftar penanda berisi penanda-penanda yang sudah diatur oleh admin dan dikeluarkan urut secara alfabet dari nama penandanya, dan jika item penanda pada daftar dipencet maka

daftar penanda akan ditutup dan kamera akan diarahkan ke lokasi penanda. Jika pengguna adalah admin, maka terdapat empat tombol tambahan untuk menambah penanda pada posisi sekarang, untuk menambah penanda pada lokasi yang dipilih, mengedit properti penanda, dan menghapus penanda. Setelah tombol untuk menambah penanda atau mengedit penanda dipencet maka akan ditampilkan panel properti penanda beserta tombol untuk menyimpan maupun membatalkannya, tetapi jika tombol untuk menghapus penanda yang dipencet maka akan keluar pemberitahuan untuk memilih penanda yang sama lagi dalam waktu satu detik, jika penanda yang sama dipilih sebelum 1 detik maka penanda akan dihapus.

Jika pengguna memilih penanda pada peta, akan dicek apakah pengguna berada pada sekitar penanda yang dipilih atau tidak, jika tidak maka tidak terjadi apa-apa, jika ya maka akan dikeluarkan panel *loading bar* untuk menunjukkan statusnya. Setelah *loading bar* sudah penuh (sampai paling kanan) maka adegan peta ditutup dan digantikan dengan adegan AR.

#### 5.1.4 Pengujian Adegan *Image Tracking*

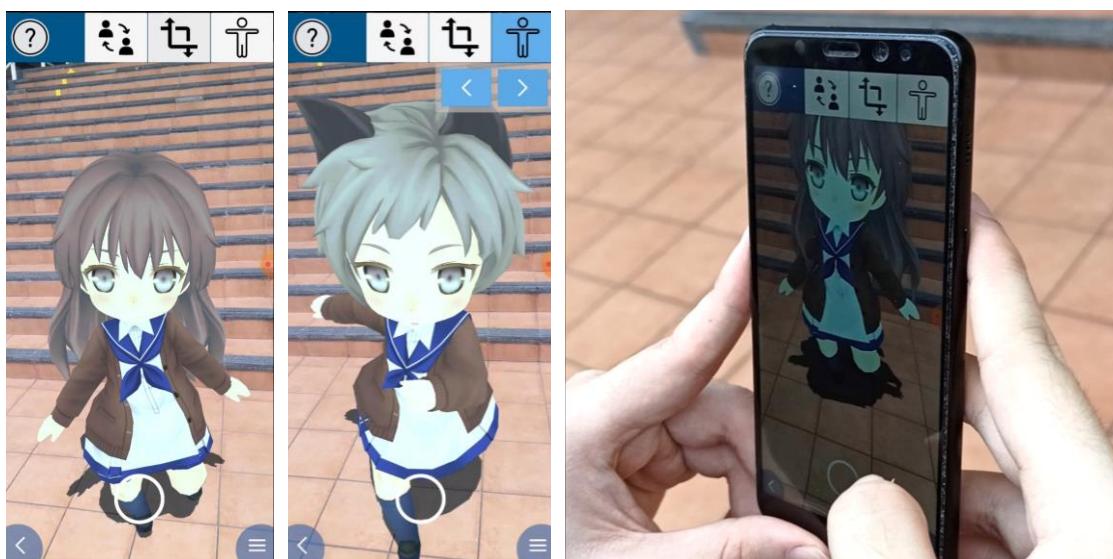
Pada adegan *image tracking*, pengguna dapat kembali ke adegan peta, atau dapat mengarahkan kamera ke gambar logo UK Petra untuk mengeluarkan tombol “Go”, dan ketika logo ditekan maka akan dikeluarkan panel *loading bar* untuk menunjukkan statusnya. Setelah *loading bar* sudah penuh (sampai paling kanan) maka adegan *image tracking* ditutup dan digantikan dengan adegan AR.



Gambar 5.9 Pengujian adegan *image tracking*

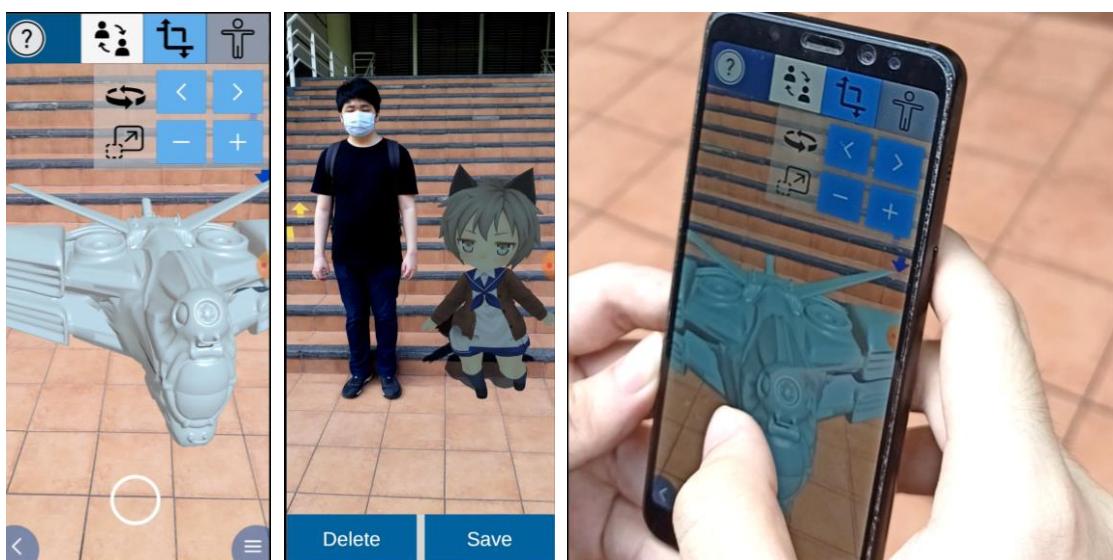
### 5.1.5 Pengujian Adegan AR

Pada adegan AR, pengguna dapat melihat penjelasan adegan AR, memilih model objek 3D, kembali ke adegan peta, mengambil gambar, membuka menu, dan mengarahkan kamera ke permukaan datar horizontal untuk pelacakan permukaan.



Gambar 5.10 Ganti karakter dan animasi pada adegan AR

Pengguna dapat memilih untuk mengganti model objek 3D virtual menjadi karakter lain atau melakukan *import file* dengan ekstensi OBJ sendiri. Jika pengguna menekan permukaan yang sudah dilacak maka objek 3D akan dikeluarkan pada posisi yang ditekan. Setelah karakter dikeluarkan maka tombol untuk mengganti animasi, mengganti rotasi pada sumbu y (*yaw*), dan mengganti skala dapat ditekan.



Gambar 5.11 Import dan pengambilan foto pada adegan AR

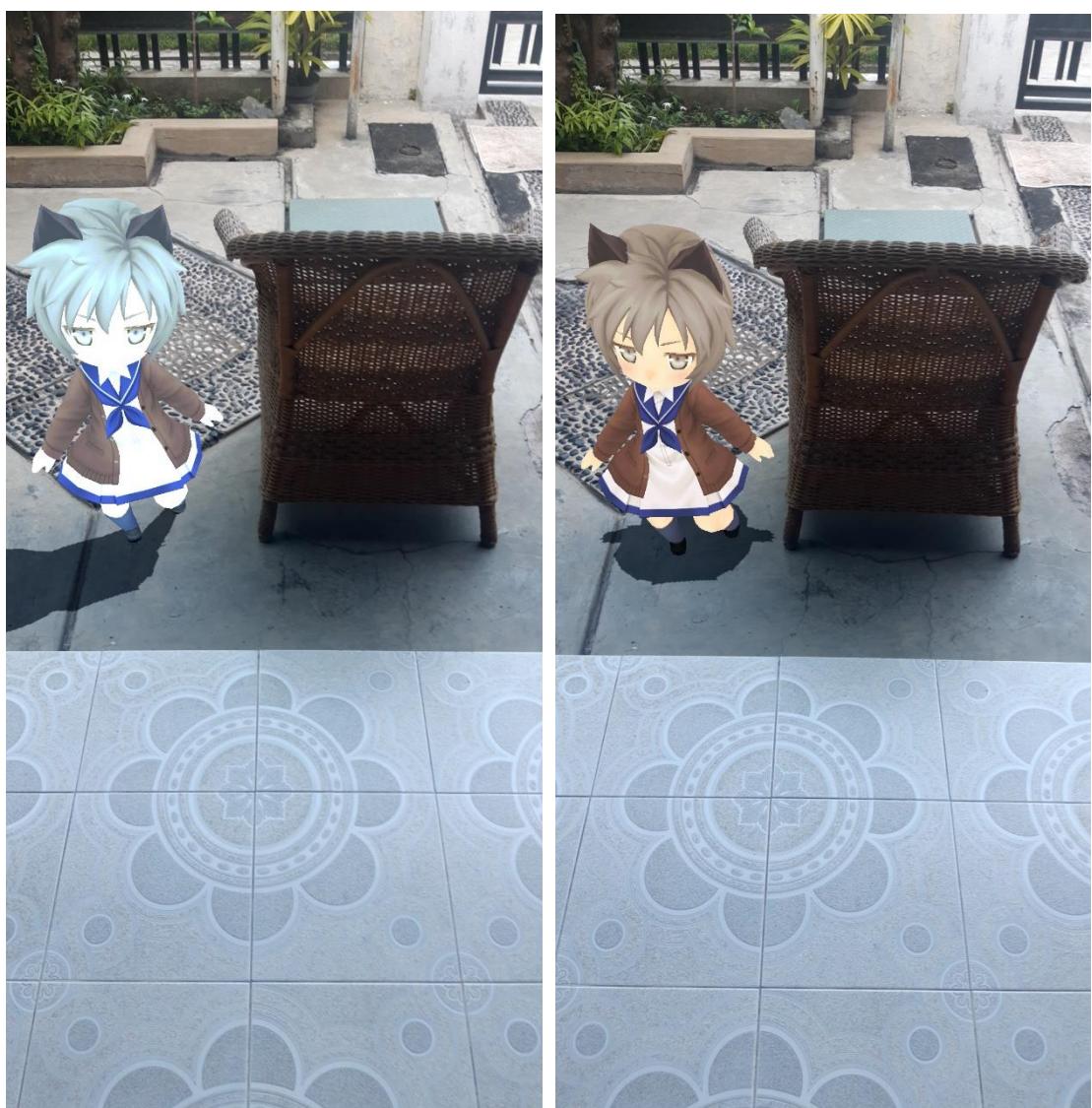
Jika pengguna ingin mengambil foto, pengguna dapat menekan tombol untuk mengambil gambar, setelah itu UI akan dimatikan, lalu gambar akan diambil dan ditampilkan pada panel *preview* gambar. Pada panel ini pengguna dapat memilih untuk membuang gambar atau menyimpannya pada galeri perangkat.

Menu pada adegan AR memiliki empat pilihan , yaitu pilihan untuk melanjutkan adegan AR, melakukan reset adegan AR, membuka panel pengaturan, dan keluar dari aplikasi. Pada panel pengaturan terdapat pilihan untuk menyalakan atau mematikan estimasi cahaya, karakter debug, dan mode debug, selain itu juga terdapat tombol untuk menutup panel pengaturan dan kembali ke menu. Jika tombol untuk keluar dari aplikasi ditekan maka panel menu akan digantikan dengan panel konfirmasi, jika pengguna memilih untuk batal maka panel konfirmasi akan dikembalikan menjadi panel menu, dan jika pengguna memilih untuk keluar maka aplikasi akan ditutup.

## 5.2 Pengujian Metode Estimasi Cahaya

Pengujian metode estimasi cahaya dibagi menjadi dua pengujian, yaitu pengujian arah bayangan yang dilakukan dengan tujuan untuk melihat apakah realistik atau tidaknya arah bayangan yang dihasilkan oleh *rendering* objek 3D virtual pada metode estimasi cahaya, dan pengujian kekurangan metode estimasi cahaya yang dilakukan dengan tujuan untuk melihat batasan-batasan yang terdapat pada metode estimasi cahaya. Pengujian arah bayangan dilakukan dengan cara mengukur sudut yang dibentuk oleh bayangan yang dihasilkan oleh objek 3D virtual dengan arah bayangan dari objek nyata pada lingkungannya, lalu dibandingkan dengan arah bayangan yang dihasilkan oleh *rendering* objek 3D virtual tanpa metode estimasi cahaya. Pengukuran sudut dilakukan dengan menggunakan “Online Protractor” ([https://www.ginifab.com/feeds/angle\\_measurement/](https://www.ginifab.com/feeds/angle_measurement/)). Pengujian metode estimasi cahaya dilakukan menggunakan perangkat pertama (Samsung Galaxy A8) dengan spesifikasi yang tertulis pada bagian awal bab 5.

### 5.2.1 Pengujian Perbandingan Arah Bayangan *Outdoor*



Gambar 5.12 Perbandingan arah bayangan *outdoor* pada siang hari (cahaya dari kanan belakang objek)

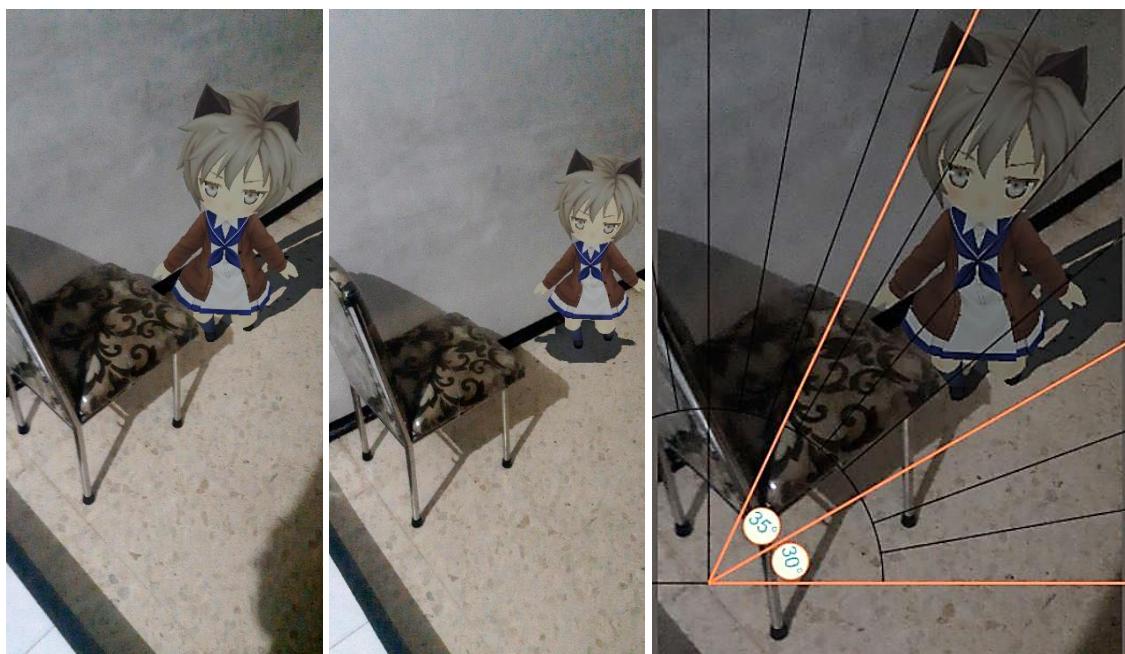
Pada lokasi *outdoor*, jika terdapat cahaya yang cukup terang dari segala arah, bayangan yang dihasilkan bisa lebih dari satu, yaitu *soft shadow* yang memiliki arah global dari posisi matahari dan *hard shadow* yang terlihat seperti tepat berada pada bawah objek. Pada metode estimasi cahaya, bayangan yang dihasilkan hanya satu *hard shadow* yang memiliki arah global, sehingga kurang realistik jika dibandingkan dengan hasil foto tanpa metode estimasi cahaya yang menghasilkan satu *hard shadow* tepat di bawah objek.



Gambar 5.13 Perbandingan arah bayangan outdoor pada malam hari (cahaya dari belakang objek)

Pada lokasi *outdoor* di malam hari, lingkungan yang dihasilkan hampir sama dengan lingkungan *indoor* karena tidak ada matahari sebagai sumber cahaya, oleh karena itu sumber cahaya yang ada hanya berasal dari lampu yang berada di sekitar lingkungan, sehingga hasil foto yang didapatkan dengan metode estimasi cahaya lebih realistik jika dibandingkan dengan hasil foto tanpa metode estimasi cahaya. Oleh karena itu, penggunaan metode estimasi cahaya kurang cocok untuk digunakan pada lokasi *outdoor* yang menghasilkan lebih dari satu bayangan seperti pada siang hari, tetapi dapat digunakan jika bayangan yang dihasilkan tidak lebih dari satu seperti pada malam hari dengan satu sumber cahaya.

### 5.2.2 Pengujian Perbandingan Arah Bayangan *Indoor*



Gambar 5.14 Perbandingan arah bayangan *indoor* pertama (cahaya dari depan kiri objek)



Gambar 5.15 Perbandingan arah bayangan *indoor* kedua (cahaya dari belakang objek)

Dapat dilihat dari gambar di atas bahwa pada metode estimasi cahaya, sudut yang dihasilkan oleh bayangan objek 3D virtual dan objek nyata pada lingkungannya cukup kecil (sekitar  $33^\circ$ ) dan searah, sehingga terlihat lebih realistik dibandingkan dengan bayangan yang dihasilkan tanpa menggunakan metode estimasi cahaya, di mana sumber cahayanya masih diasumsikan selalu berasal dari atas objek dan memiliki arah yang selalu ke bawah sehingga

bayangan yang dihasilkan selalu tepat di bawah objek. Jenis bayangan yang digunakan pada penelitian ini adalah *hard shadows*, karena pencahayaan yang digunakan adalah *directional light* (jarak tidak berpengaruh) dan material permukaan (properti) tidak diketahui, sehingga penggunaan *soft shadow* tidak terlalu berpengaruh dan hanya menambah penggunaan *resource*.

Untuk mengetahui dengan lebih jelas apakah arah bayangan yang dihasilkan oleh karakter 3D virtual realistik atau tidak menurut pendapat berbagai orang, maka akan digunakan survei untuk pengukuran realistiknya intensitas cahaya dan hasil bayangan dari satu (sangat tidak realistik) sampai lima (sangat realistik). Pengujian yang digunakan sebagai perbandingan pada survei adalah kedua pengujian pada lingkungan *outdoor* (pada siang hari dan malam hari), dan enam pengujian pada lingkungan *indoor* yang sudah ditunjukkan (kedua pengujian *indoor* diatas) dan akan ditunjukkan di bawah ini. Survei dilakukan dengan menggunakan “Google Forms” (<https://docs.google.com/forms/>) pada :

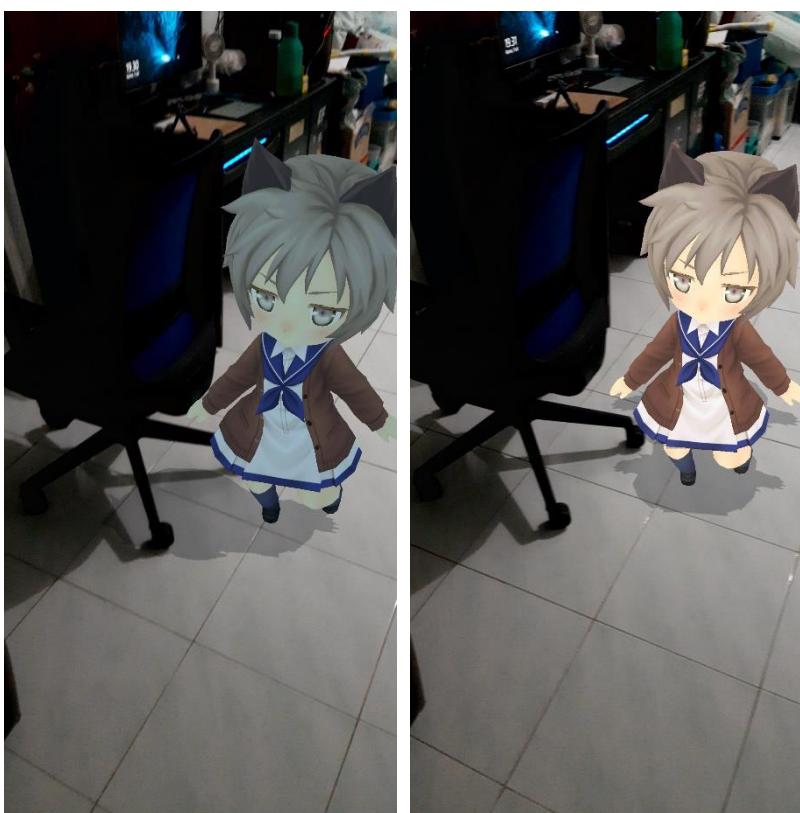
[“https://docs.google.com/forms/d/e/1FAIpQLSdcjYLNahqe0nx6M8MGqxmeB8ckNDwBx4\\_bz8hBRat0kP1CMA/viewform?usp=sf\\_link”](https://docs.google.com/forms/d/e/1FAIpQLSdcjYLNahqe0nx6M8MGqxmeB8ckNDwBx4_bz8hBRat0kP1CMA/viewform?usp=sf_link)



Gambar 5.16 Perbandingan arah bayangan *indoor* ketiga (cahaya dari atas objek)



Gambar 5.17 Perbandingan arah bayangan *indoor* keempat (cahaya dari atas objek)

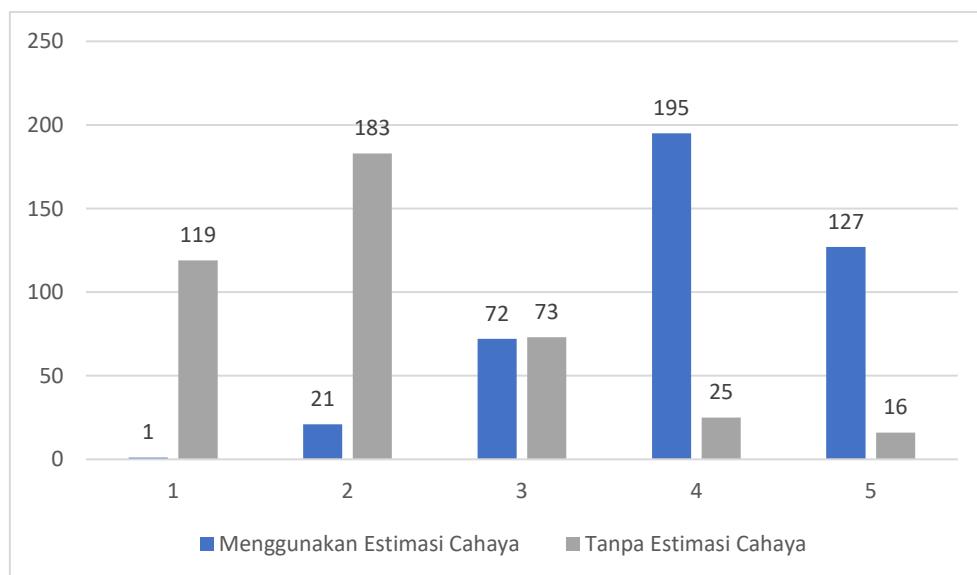


Gambar 5.18 Perbandingan arah bayangan *indoor* kelima (cahaya dari belakang kanan objek)



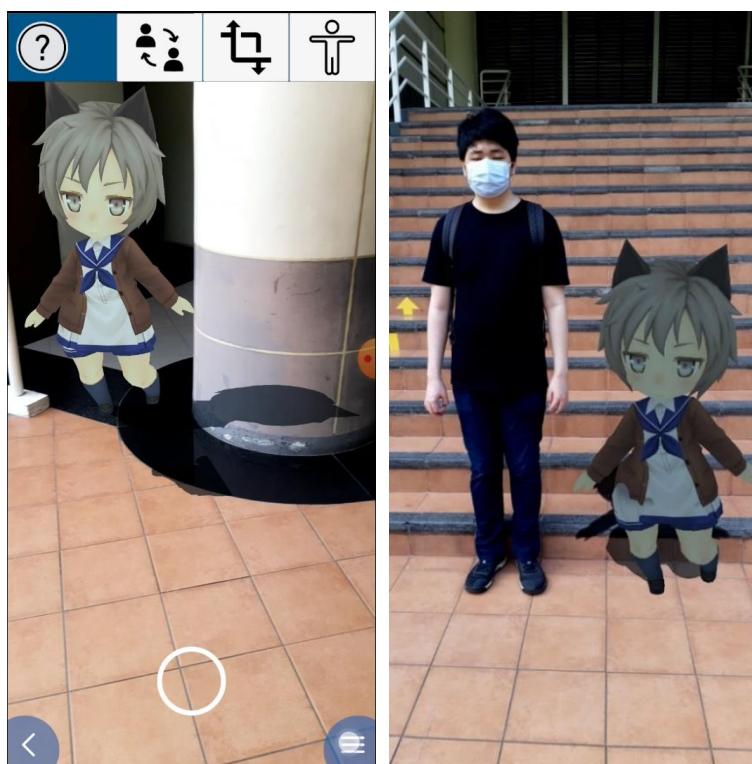
Gambar 5.19 Perbandingan arah bayangan *indoor* keenam (cahaya dari depan kanan objek)

Hasil survei pada lampiran 1 menunjukkan bahwa menurut 52 responden yang sudah mengikuti survei, foto yang menggunakan estimasi cahaya memiliki nilai realistik 4.026 dari 5 poin dan foto yang tidak menggunakan estimasi cahaya memiliki nilai realistik 2.125 dari 5 poin. Oleh karena itu, dapat disimpulkan bahwa foto yang menggunakan estimasi cahaya lebih realistik jika dibandingkan dengan foto yang tidak menggunakan estimasi cahaya.



Gambar 5.20 Perbandingan nilai realistik

### 5.2.3 Kekurangan Lain dari Metode Estimasi Cahaya



Gambar 5.21 Pengujian kekurangan lain dari metode estimasi cahaya

Dapat dilihat dari gambar di atas bahwa metode estimasi cahaya tidak dapat membuat bayangan pada bidang vertikal, dan tidak dapat membedakan pakaian dengan lingkungan. Pakaian hitam dianggap lingkungan gelap sehingga intensitas cahaya diturunkan karena Intensitas dari cahaya didapatkan dengan cara memperhitungkan rata-rata dari kecerahan keseluruhan gambar yang diproses.



Gambar 5.22 Pengambilan foto pada lingkungan dengan dua sumber cahaya (cahaya dari depan kanan dan kiri objek)

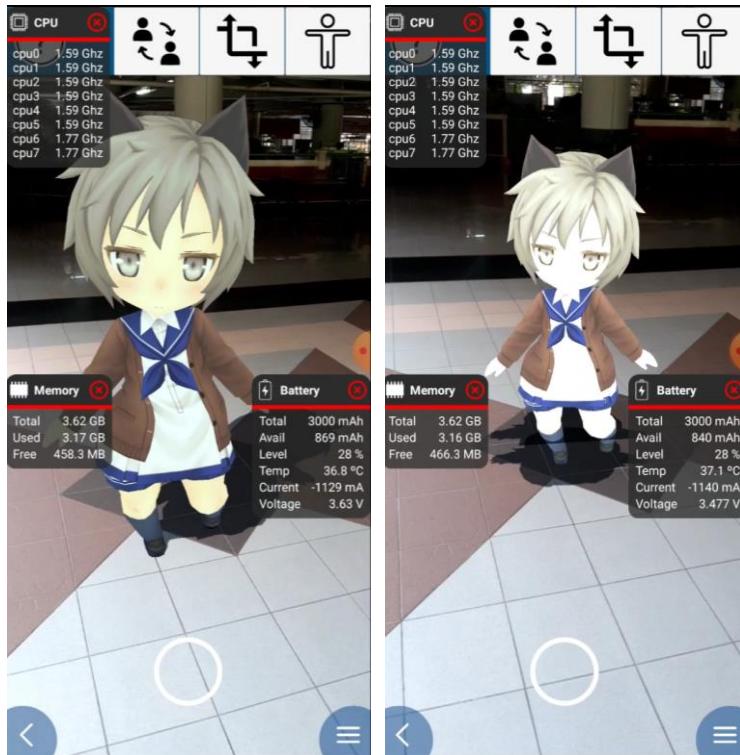
Selain itu, karena pada metode estimasi cahaya bayangan yang dihasilkan hanya satu *hard shadow* yang memiliki arah global, jika pada lingkungan tempat foto diambil terdapat lebih dari satu sumber cahaya dan menghasilkan lebih dari satu bayangan, maka arah bayangan yang dihasilkan oleh objek 3D virtual akan menjadi kurang realistik.

Oleh karena itu, berdasarkan hasil pengujian dan survei yang terkumpul, dapat disimpulkan bahwa lingkungan yang memiliki satu sumber cahaya yang cukup terang untuk menghasilkan satu bayangan *hard shadow* yang jelas adalah kondisi tempat foto yang ideal agar hasil foto yang didapatkan menjadi lebih baik.

### 5.3 Pengujian Perbedaan Penggunaan Resource

Pengujian perbedaan penggunaan *resource* dilakukan dengan tujuan untuk melihat penggunaan *resource* pada *rendering* objek 3D, yang dilakukan dengan cara membandingkan penggunaan CPU, penggunaan RAM, dan penggunaan baterai selama 30 detik, dan rata-rata waktu untuk *merender* (*Average Frames per Second*) yang stabil, diantara *rendering* yang menggunakan dan tidak menggunakan *light estimation*. Pengukuran penggunaan *resource* dilakukan dengan menggunakan “System Monitor Float Free”, yang didapatkan dari “Google Play Store” di Android. Pengujian perbedaan penggunaan *resource* dilakukan menggunakan perangkat pertama (Samsung Galaxy A8) dan kedua (Xiaomi Redmi Note 8 Pro) dengan spesifikasi yang tertulis pada bagian awal bab 5.

### 5.3.1 Pengujian Penggunaan Resource Pada Perangkat Pertama



Gambar 5.23 Penggunaan CPU, RAM, dan baterai pada perangkat pertama

Tabel 5.1

Penggunaan CPU, RAM, dan baterai pada perangkat pertama menggunakan estimasi cahaya

Detik	Low CPU (GHz)	High CPU (GHz)	RAM (GB)	Baterai (mA)
1	1.59	1.87	2.68	923
2	1.59	1.87	2.68	923
3	1.59	1.87	2.67	923
4	1.59	1.87	2.67	923
5	1.59	1.87	2.66	923
6	1.59	1.87	2.66	923
7	1.59	1.87	2.66	923
8	1.59	1.87	2.65	923
9	1.59	1.87	2.66	923
10	1.59	1.87	2.66	923

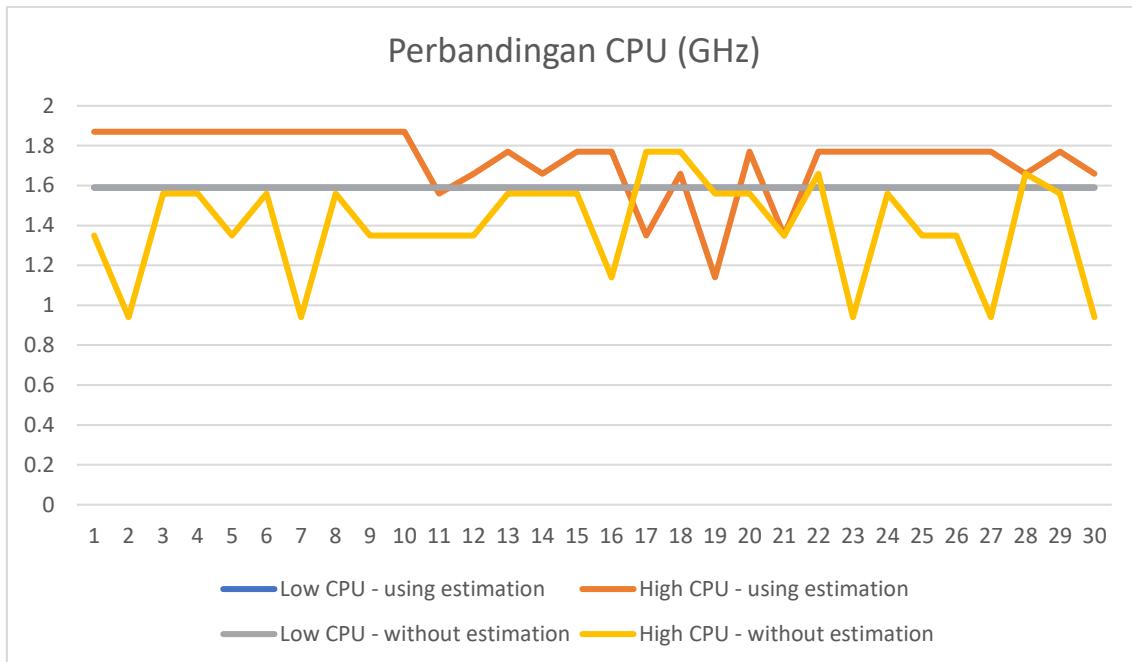
11	1.59	1.56	2.65	923
12	1.59	1.66	2.65	923
13	1.59	1.77	2.65	923
14	1.59	1.66	2.65	923
15	1.59	1.77	2.64	923
16	1.59	1.77	2.64	923
17	1.59	1.35	2.64	923
18	1.59	1.66	2.64	923
19	1.59	1.14	2.64	923
20	1.59	1.77	2.65	923
21	1.59	1.35	2.65	923
22	1.59	1.77	2.65	923
23	1.59	1.77	2.65	923
24	1.59	1.77	2.65	923
25	1.59	1.77	2.65	923
26	1.59	1.77	2.65	923
27	1.59	1.77	2.64	923
28	1.59	1.66	2.65	923
29	1.59	1.77	2.65	923
30	1.59	1.66	2.65	1318
Rata-rata	1.59	1.73	2.65	936.17

Tabel 5.2

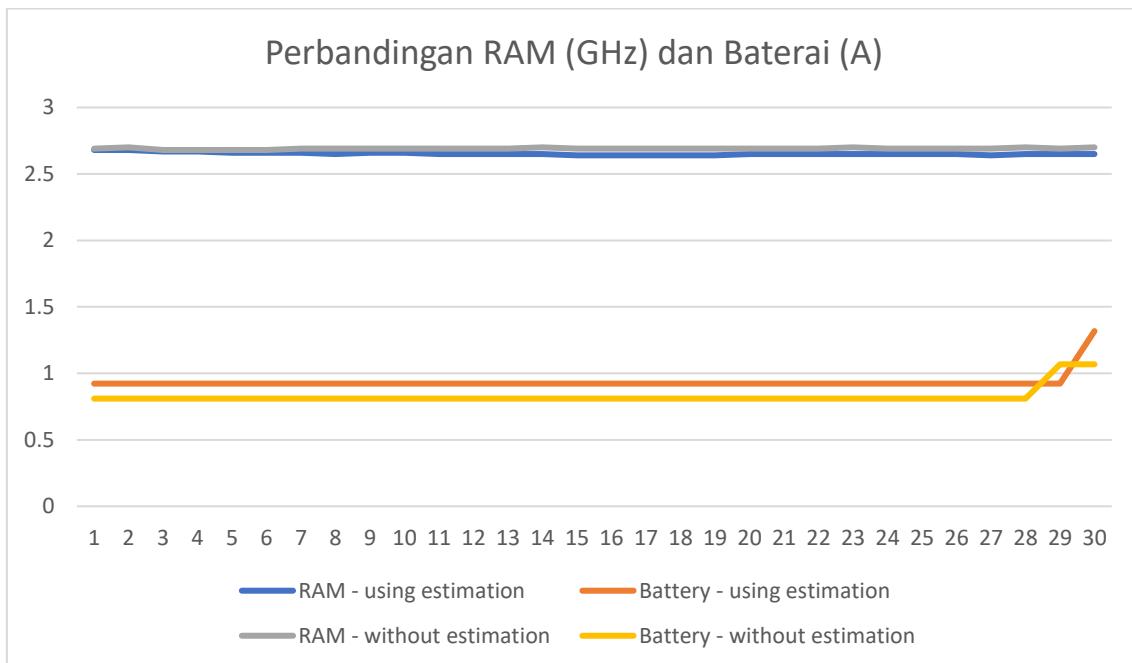
Penggunaan CPU, RAM, dan baterai pada perangkat pertama tanpa estimasi cahaya

Detik	Low CPU (GHz)	High CPU (GHz)	RAM (GB)	Baterai (mA)
1	1.59	1.35	2.69	810
2	1.59	0.94	2.7	810
3	1.59	1.56	2.68	810
4	1.59	1.56	2.68	810
5	1.59	1.35	2.68	810
6	1.59	1.56	2.68	810
7	1.59	0.94	2.69	810

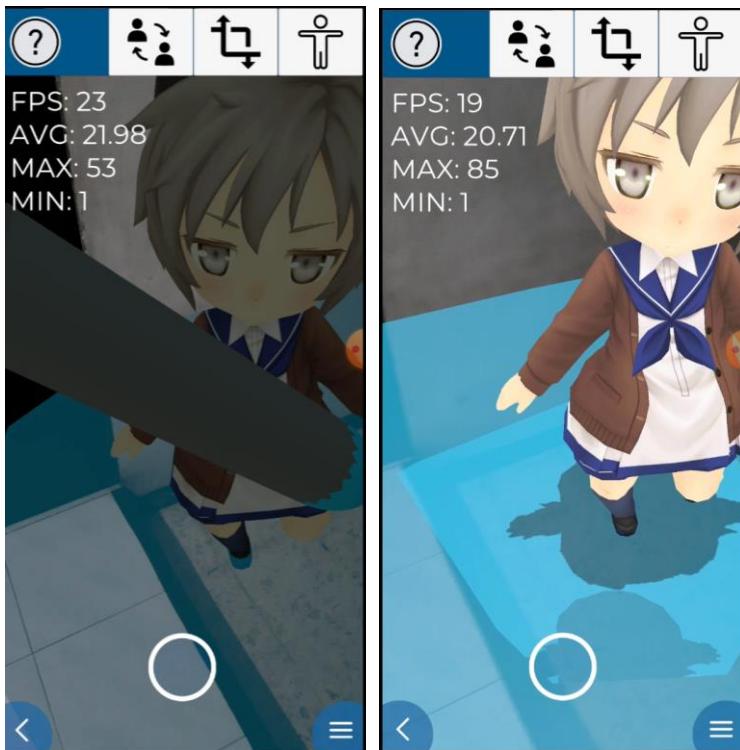
8	1.59	1.56	2.69	810
9	1.59	1.35	2.69	810
10	1.59	1.35	2.69	810
11	1.59	1.35	2.69	810
12	1.59	1.35	2.69	810
13	1.59	1.56	2.69	810
14	1.59	1.56	2.70	810
15	1.59	1.56	2.69	810
16	1.59	1.14	2.69	810
17	1.59	1.77	2.69	810
18	1.59	1.77	2.69	810
19	1.59	1.56	2.69	810
20	1.59	1.56	2.69	810
21	1.59	1.35	2.69	810
22	1.59	1.66	2.69	810
23	1.59	0.94	2.7	810
24	1.59	1.56	2.69	810
25	1.59	1.35	2.69	810
26	1.59	1.35	2.69	810
27	1.59	0.94	2.69	810
28	1.59	1.66	2.7	810
29	1.59	1.56	2.69	1068
30	1.59	0.94	2.7	1068
Rata-rata	1.59	1.40	2.69	827.20



Gambar 5.24 Perbandingan penggunaan CPU pada perangkat pertama



Gambar 5.25 Perbandingan penggunaan RAM dan baterai pada perangkat pertama



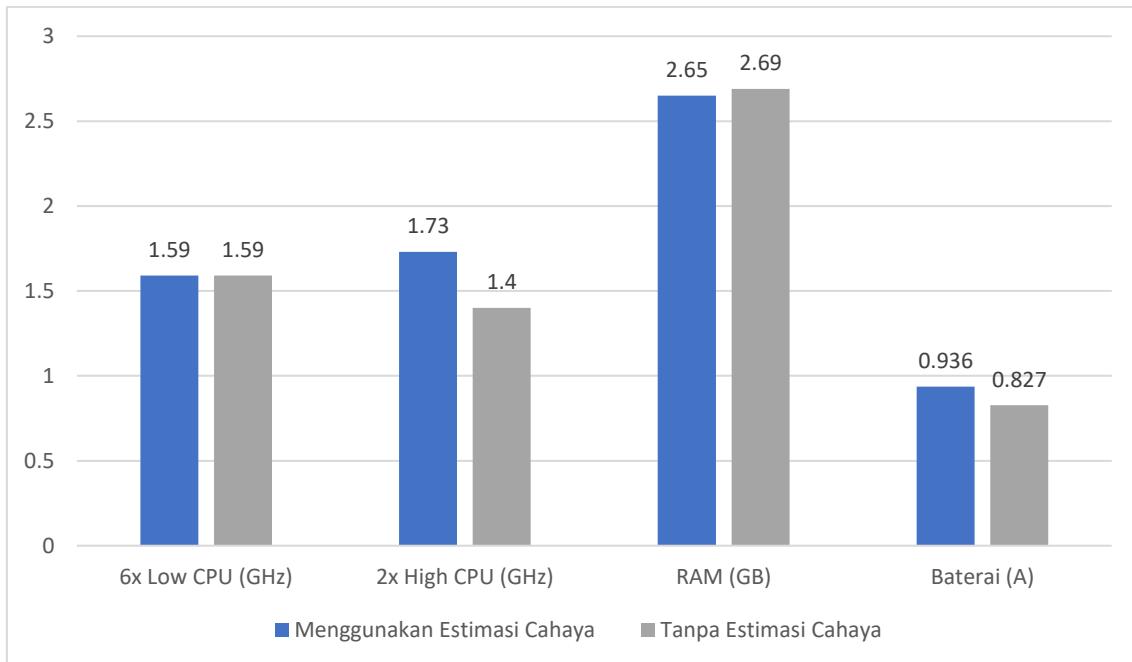
Gambar 5.26 Pengujian FPS pada perangkat pertama

Tabel 5.3

Rata-rata resource dan FPS pada perangkat pertama (pembulatan 2 angka di belakang koma)

	Menggunakan estimasi cahaya	Tanpa estimasi cahaya	Persentase Peningkatan
6x Low CPU	1.59 GHz	1.59 GHz	0.00%
2x High CPU	1.73 GHz	1.40 GHz	23.53%
RAM	2.65 GB	2.69 GB	-1.39%
Baterai	936.17 mA	827.20 mA	13.17%
FPS	21.98	20.71	6.13%

Untuk membandingkan nilai pada tabel “Rata-rata resource dan FPS pada perangkat pertama” dengan lebih jelas, Berikut ini adalah grafik batang yang dibuat untuk perbandingan nilai.



Gambar 5.27 Perbandingan nilai rata-rata *resource* dan FPS pada perangkat pertama

### 5.3.2 Pengujian Penggunaan Resource Pada Perangkat Kedua



Gambar 5.28 Penggunaan CPU, RAM, dan baterai pada perangkat kedua

Tabel 5.4

Penggunaan CPU, RAM, dan baterai pada perangkat kedua menggunakan estimasi cahaya

Detik	Low CPU (GHz)	High CPU (GHz)	RAM (GB)	Baterai (mA)
1	1.28	2.05	4.14	1367
2	1.08	2.05	4.21	997
3	2.00	2.05	4.19	1696
4	2.00	1.99	4.16	1136
5	2.00	2.05	4.17	1411
6	2.00	1.92	4.18	1611
7	1.80	2.05	4.19	1417
8	1.80	1.31	4.19	1246
9	2.00	2.05	4.19	1533
10	1.18	0.92	4.18	957
11	1.18	1.92	4.18	1343
12	1.62	0.92	4.15	1132
13	1.87	1.80	4.16	1195
14	2.00	2.05	4.19	1369
15	1.62	1.80	4.18	1192
16	2.00	2.05	4.17	1452
17	1.28	1.86	4.17	1332
18	2.00	2.05	4.17	1256
19	0.98	0.92	4.17	945
20	2.00	1.92	4.16	1085
21	1.87	1.92	4.16	1036
22	1.38	0.92	4.16	1091
23	0.88	0.92	4.16	968
24	2.00	2.05	4.16	1166
25	1.87	1.92	4.16	1059
26	2.00	1.17	4.16	917
27	1.62	1.17	4.16	1543
28	2.00	1.00	4.14	1273
29	1.80	1.00	4.14	1092

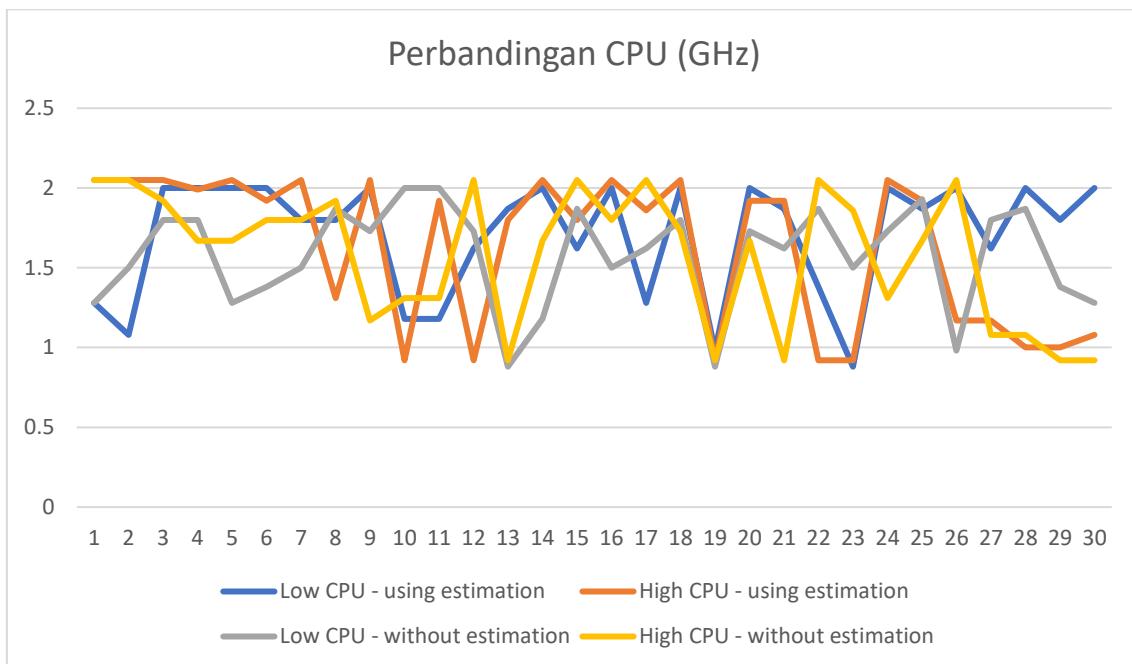
30	2.00	1.08	4.14	1409
Rata-rata	1.70	1.63	4.17	1240.87

Tabel 5.5

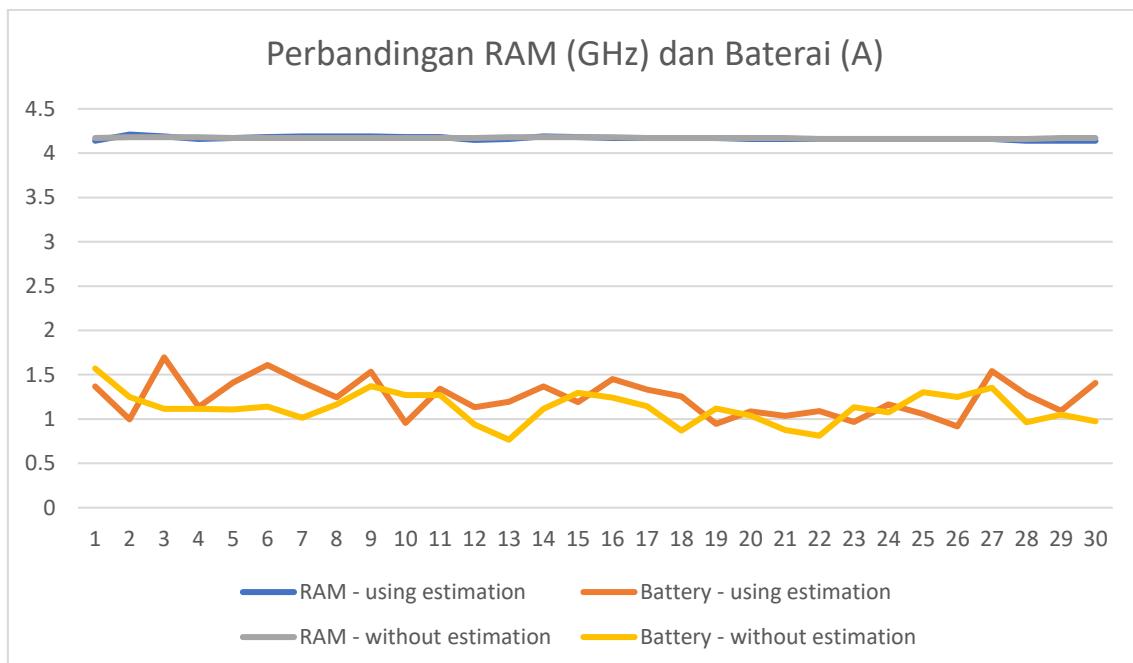
Penggunaan CPU, RAM, dan baterai pada perangkat kedua tanpa estimasi cahaya

Detik	Low CPU (GHz)	High CPU (GHz)	RAM (GB)	Baterai (mA)
1	1.28	2.05	4.17	1571
2	1.5	2.05	4.18	1250
3	1.80	1.92	4.18	1116
4	1.80	1.67	4.18	1116
5	1.28	1.67	4.17	1107
6	1.38	1.80	4.17	1141
7	1.50	1.80	4.17	1015
8	1.87	1.92	4.17	1166
9	1.73	1.17	4.17	1372
10	2.00	1.31	4.17	1270
11	2.00	1.31	4.17	1270
12	1.73	2.05	4.17	938
13	0.88	0.92	4.18	767
14	1.18	1.67	4.18	1115
15	1.87	2.05	4.18	1297
16	1.50	1.80	4.18	1242
17	1.62	2.05	4.17	1143
18	1.80	1.73	4.17	870
19	0.88	0.92	4.17	1118
20	1.73	1.67	4.17	1041
21	1.62	0.92	4.17	878
22	1.87	2.05	4.16	810
23	1.50	1.86	4.16	1134
24	1.73	1.31	4.16	1077
25	1.93	1.67	4.16	1304
26	0.98	2.05	4.16	1250

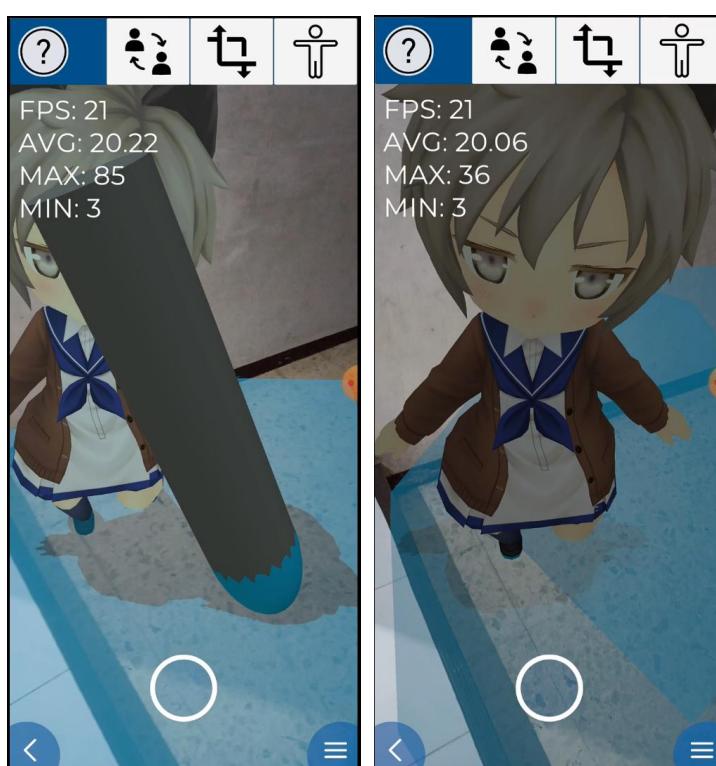
27	1.80	1.08	4.16	1354
28	1.87	1.08	4.16	965
29	1.38	0.92	4.17	1050
30	1.28	0.92	4.17	976
Rata-rata	1.57	1.58	4.17	1124.10



Gambar 5.29 Perbandingan penggunaan CPU pada perangkat kedua



Gambar 5.30 Perbandingan penggunaan RAM dan baterai pada perangkat kedua



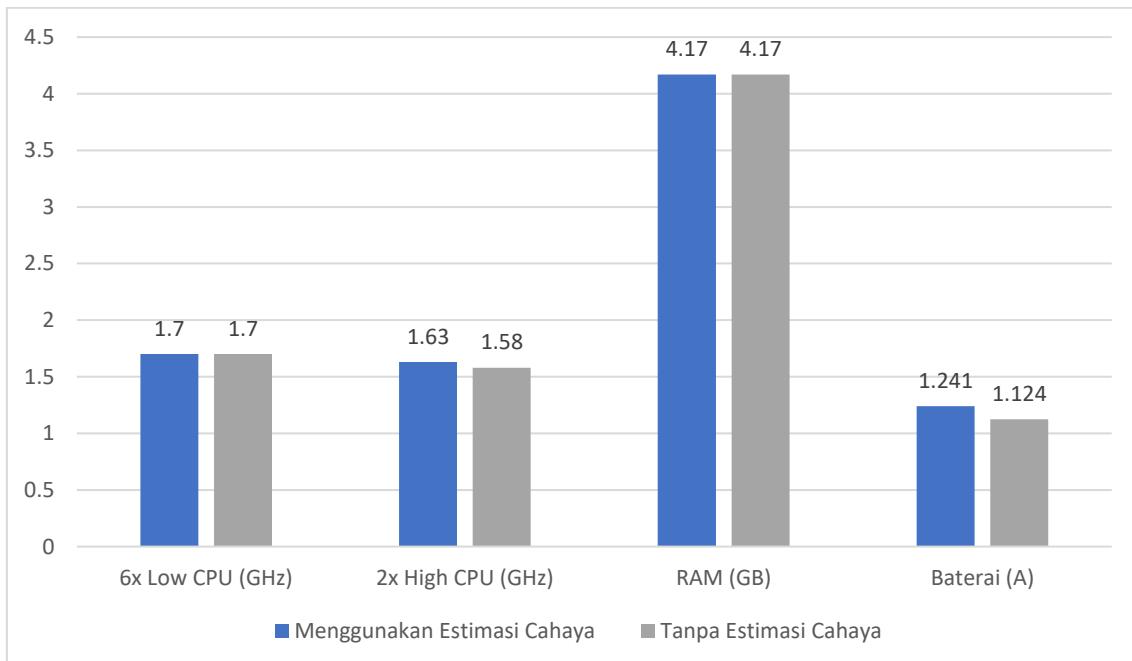
Gambar 5.31 Pengujian FPS pada perangkat kedua

Tabel 5.6

Rata-rata *resource* dan FPS pada perangkat kedua (pembulatan 2 angka di belakang koma)

	Menggunakan estimasi cahaya	Tanpa estimasi cahaya	Persentase Peningkatan
6x Low CPU	1.70 GHz	1.57 GHz	8.46%
2x High CPU	1.63 GHz	1.58 GHz	3.14%
RAM	4.17 GB	4.17 GB	-0.05%
Baterai	1240.87 mA	1124.10 mA	10.39%
FPS	20.22	20.06	-0.20%

Untuk membandingkan nilai pada tabel “Rata-rata resource dan FPS pada perangkat kedua” dengan lebih jelas, Berikut ini adalah grafik batang yang dibuat untuk perbandingan nilai.

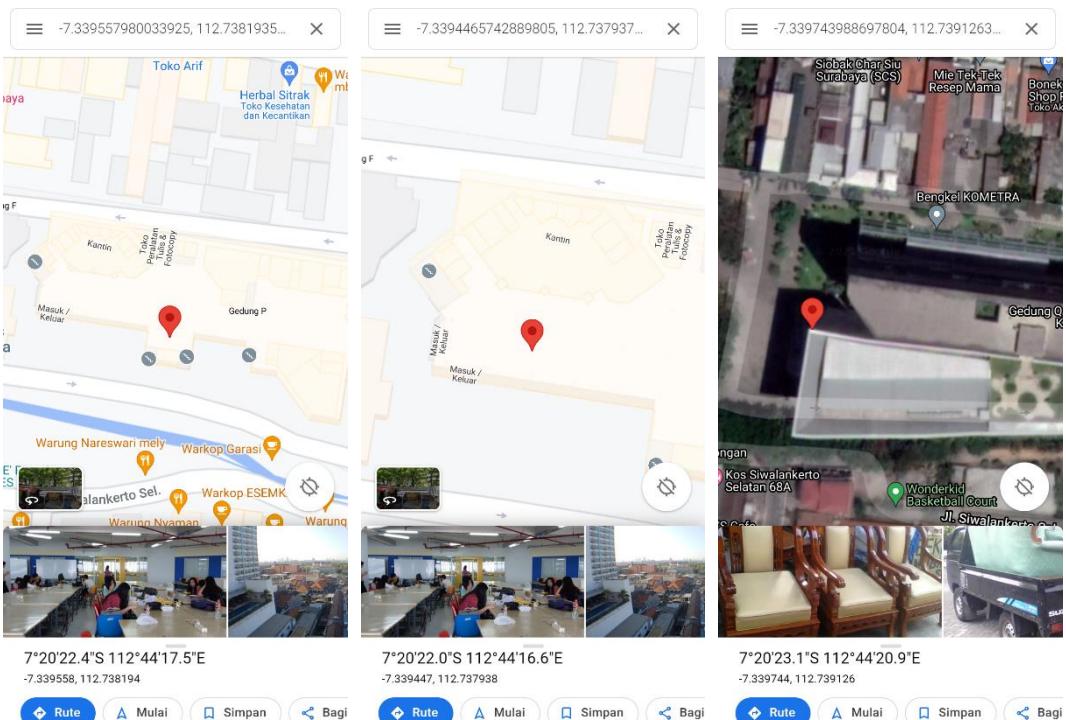


Gambar 5.32 Perbandingan nilai rata-rata *resource* dan FPS pada perangkat kedua

Dari hasil pengujian penggunaan *resource* yang dilakukan, dapat dilihat bahwa penggunaan RAM dan FPS kurang lebih sama, dan perbedaan penggunaan *resource* lainnya antara metode estimasi cahaya dengan penggunaan *resource* tanpa metode estimasi cahaya cukup kecil untuk tetap dapat digunakan pada perangkat *mobile*. Tetapi durasi waktu penggunaan yang berkepanjangan dalam suatu sesi akan meningkatkan temperatur baterai dan membuat sistem operasi untuk mempengaruhi penggunaan *resource* dalam pencegahan *Overheating*.

## 5.4 Pengujian Akurasi Pelacakan Lokasi

Pengujian akurasi pelacakan lokasi dilakukan dengan tujuan untuk melihat seberapa akuratnya pelacakan lokasi pada suatu perangkat, yang dilakukan dengan cara mengukur rata-rata jarak antara lokasi penanda yang dikeluarkan di perangkat pengguna berdasarkan sensor lokasi dengan lokasi di mana seharusnya penanda dikeluarkan.



Gambar 5.33 Lokasi pengujian

Pengujian akurasi pelacakan lokasi dilakukan pada 3 lokasi, yaitu “Kolam Gedung P”, “Selasar Gedung P”, dan “Depan Gedung Q”. Informasi lokasi seharusnya penanda dikeluarkan didapatkan dari “Google Maps” (<https://www.google.com/maps>), dan perhitungan selisih jarak didapatkan dari “Movable Type Scripts” (<https://www.movable-type.co.uk/scripts/latlong.html>). Pengujian akurasi pelacakan lokasi dilakukan menggunakan perangkat pertama (Samsung Galaxy A8) dan kedua (Xiaomi Redmi Note 8 Pro) dengan spesifikasi yang tertulis pada bagian awal bab 5.

### 5.4.1 Pengujian Akurasi Perangkat Pertama

Tabel 5.7

Akurasi lokasi yang terlacak pada perangkat pertama

Nomor	Lokasi Terlacak	Lokasi Sesungguhnya	Akurasi / Jarak (Meter)
1	-7.3394690, 112.7381780	Kolam Gedung P -7.339557980033925, 112.7381935716804	10.04
2	-7.3394991, 112.7381584		7.610
3	-7.3395509, 112.7381842		1.299
4	-7.3395876, 112.7381751		3.873
5	-7.3395581, 112.7381741		2.147
6	-7.3394340, 112.7379281	Selasar Gedung P -7.3394465742889805, 112.73793797415902	1.772
7	-7.3394316, 112.7379606		3
8	-7.3394472, 112.7379627		2.728
9	-7.3394534, 112.7379468		1.234
10	-7.3394657, 112.7379399		2.137
11	-7.3397582, 112.7391951	Depan Gedung Q -7.339743988697804, 112.7391263175496	7.748
12	-7.3397591, 112.7391562		3.699
13	-7.3397475, 112.7391071		2.155
14	-7.3397376, 112.7391189		1.083
15	-7.3397402, 112.7391287		0.4965

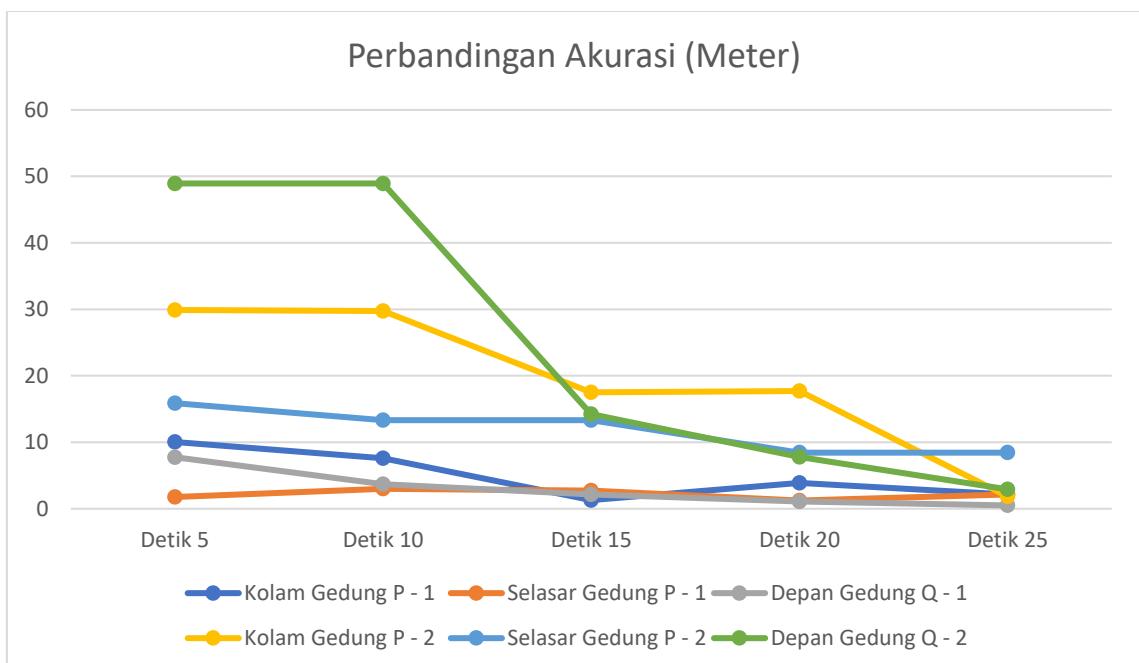
#### 5.4.2 Pengujian Akurasi Perangkat Kedua

Tabel 5.8

Akurasi lokasi yang terlacak pada perangkat kedua

Nomor	Lokasi Terlacak	Lokasi Sesungguhnya	Akurasi / Jarak (Meter)
1	-7.3395083, 112.7384600	Kolam Gedung P -7.339557980033925, 112.7381935716804	29.90
2	-7.3392933, 112.7382317		29.73
3	-7.3394400, 112.7380883		17.52
4	-7.3395233, 112.7380367		17.72
5	-7.3397233, 112.7382033		1.842
6	-7.3395383, 112.7380483	Selasar Gedung P -7.3394465742889805, 112.73793797415902	15.88
7	-7.3393600, 112.7380217		13.34
8	-7.3393600, 112.7380217		13.34
9	-7.3394167, 112.7380083		8.437
10	-7.3394167, 112.7380083		8.437
11	-7.3395600, 112.7387233	Depan Gedung Q	48.93

12	-7.3395600, 112.7387233	-7.339743988697804, 112.7391263175496	48.93
13	-7.3398683, 112.7391567		14.22
14	-7.3397783, 112.7390650		7.764
15	-7.3397467, 112.7391000		2.918



Gambar 5.34 Perbandingan akurasi pada perangkat pertama dan kedua

Kedua pengujian di atas dilakukan pada 3 lokasi yang berbeda, dan jarak durasi di antara masing-masing pelacakan sekitar 5 detik. Dapat dilihat dari kedua pengujian akurasi perangkat bahwa rata-rata akurasi pada detik ke 25 cukup kecil untuk perangkat pertama (1.59 Meter) dan perangkat kedua (4.4 Meter), tetapi perangkat kedua membutuhkan waktu lebih lama untuk mendapatkan hasil yang lebih akurat. Dari hasil pengujian yang telah dilakukan dapat disimpulkan bahwa penggunaan GPS sebagai sumber informasi lokasi pada aplikasi berbasis lokasi bergantung pada perangkat yang digunakan, oleh karena itu, digunakan pelacakan *image tracking* sebagai cadangan pelacakan untuk menentukan lokasi pengguna. Selain itu, dari pengujian ini dapat dilihat bahwa semakin lama suatu perangkat melacak lokasi pada suatu tempat tertentu, maka rata-rata pelacakan lokasinya menjadi semakin akurat.

## 6. KESIMPULAN DAN SARAN

Pada bab ini memberikan penjelasan tentang kesimpulan yang diperoleh dalam aplikasi untuk *rendering* karakter 3D virtual secara *real-time* menggunakan metode *light estimation* pada *augmented reality* berbasis lokasi dan sejumlah saran untuk pengembangan lebih lanjut.

### 6.1 Kesimpulan

Dari hasil pembuatan sistem beserta pengujian dan analisis yang sudah dilakukan pada bab 5, maka dapat diperoleh beberapa kesimpulan sebagai berikut :

- Penggunaan fitur pada aplikasi sudah berjalan dengan baik karena semua fitur sudah dapat dijalankan secara *real-time* dari awal sampai pengambilan foto.
- Kondisi lingkungan lokasi tempat foto yang disarankan agar aplikasi dapat digunakan dengan baik adalah lingkungan yang menghasilkan satu bayangan *hard shadow* (lingkungan dengan satu sumber cahaya yang cukup terang), karena aplikasi kurang cocok digunakan pada lingkungan *outdoor* yang menghasilkan lebih dari satu bayangan seperti pada siang hari.
- Arah bayangan yang dihasilkan pada lingkungan *indoor* cukup akurat (sekitar 33°) sehingga lebih baik daripada tanpa penggunaan metode estimasi cahaya.
- Metode estimasi cahaya memiliki kelemahan bayangan yang dihasilkan tidak dapat dihasilkan pada bidang vertikal, dan akurasi intensitas cahaya tergantung cahaya rata-rata dari gambar *input*.
- Penggunaan metode estimasi cahaya cukup ringan untuk digunakan pada perangkat *mobile* karena FPS yang dihasilkan masih sama (sekitar 20 FPS pada perangkat yang digunakan), dan penggunaan RAM hampir sama, walaupun ada peningkatan penggunaan CPU dan baterai, tetapi cukup kecil untuk tetap dapat digunakan pada perangkat *mobile*.
- Akurasi pelacakan lokasi tergantung dari perangkat yang digunakan, akurasi rata-rata pada detik ke 25 cukup kecil pada perangkat pertama (1.59 meter), dan pada perangkat kedua (4.4 meter), tetapi perangkat kedua membutuhkan waktu yang lebih lama untuk mendapatkan hasil yang akurat, sehingga jika hasil yang didapatkan kurang akurat dapat menggunakan pelacakan cadangan berupa *image tracking*, Selain itu *image tracking* juga dapat digunakan untuk membedakan lokasi antar lantai atau ketinggian.

## **6.2 Saran**

Saran yang dapat diberikan untuk penyempurnaan dan pengembangan lebih lanjut sebagai adalah berikut :

- Membuat mode khusus untuk membedakan fotografi *indoor* maupun *outdoor*
- Menggunakan penanda khusus untuk setiap karakter yang disediakan pada aplikasi
- Dapat melakukan navigasi dari lokasi pengguna sampai ke lokasi penanda yang diinginkan
- Menggunakan objek 3D virtual berupa karakter maskot yang dibuat dan didesain sendiri oleh Universitas Kristen Petra.
- Membuat aplikasi yang dapat dijalankan pada sistem operasi iOS.

## DAFTAR REFERENSI

- Akenine-Möller, T., Haines, E., & Hoffman, N. (2018). *Real-Time Rendering*. A K Peters/CRC Press; 4th edition (August 6, 2018).
- Alha, K., Koskinen, E., Paavilainen, J., & Hamari, J. (2019). Why do people play location-based augmented reality games: A study on Pokémon GO. *Computers in Human Behavior*, 93, 114-122. doi:10.1016/j.chb.2018.12.008
- Andrea, R., Lailiyah, S., Agus, F., & Ramadiani, R. (2019). "Magic Boosed" an elementary school geometry textbook with marker-based augmented reality. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 17(3), 1242-1249. doi:10.12928/telkomnika.v17i3.11559
- Anggara, D. W., Ismail, A. W., Machfiroh, R., Budiman, A., Rahmansyah, A., Dahliyusmanto, . . . Rahim, M. S. (2020). GRayscale IMAGE ENHANCEMENT FOR ENHANCING FEATURES DETECTION IN MARKER-LESS AUGMENTED REALITY TECHNOLOGY. *Journal of Theoretical and Applied Information Technology*, 98(13), 2671-2683.
- Aromaa, S., Väätänen, A., Aaltonen, I., Goriachev, V., Helin, K., & Karjalainen, J. (2020). Awareness of the real-world environment when using augmented reality head-mounted display. *Applied Ergonomics*, 88, 103145. doi:10.1016/j.apergo.2020.103145
- Brito, P. Q., & Stoyanova, J. (2018). Marker versus Markerless Augmented Reality. Which Has More Impact on Users? *International Journal of Human–Computer Interaction*, 34(9), 819-833. doi:10.1080/10447318.2017.1393974
- Dewan Perwakilan Rakyat Republik Indonesia. (2011, April 21). *Geospasial*. Retrieved from <https://paralegal.id/pengertian/geospasial/>
- Díaz-García, J., Brunet, P., Navazo, I., & Vázquez, P.-P. (2018). Progressive ray casting for volumetric models on mobile devices. *Computers & Graphics*, 73, 1-16. doi:10.1016/j.cag.2018.02.007
- Fisher, S. (1989, August 22). OS/2 EE to Get 3270 Interface Early. *InfoWorld*, 5.
- Frahm, J.-m., Koeser, K., Grest, D., & Koch, R. (2005). Markerless augmented reality with light source estimation for direct illumination. In *Conference on Visual Media Production CVMP*, (pp. 211-220).
- Georgiou, Y., & Kyza, E. A. (2018). Relations between student motivation, immersion and learning outcomes in location-based augmented reality settings. *Computers in Human Behavior*, 89, 173-181. doi:10.1016/j.chb.2018.08.011
- Gomez-Jauregui, V., Manchado, C., Del-Castillo-Igareda, J., & Otero, C. (2019). Quantitative evaluation of overlaying discrepancies in mobile augmented reality applications for AEC/FM. *Advances in Engineering Software*, 127, 124-140. doi:10.1016/j.advengsoft.2018.11.002

- Google LLC. (2020, October 6). *Using ARCore to light models in a scene*. Retrieved from <https://developers.google.com/ar/develop/unity/light-estimation>
- Greenwald, S. (2003). *Spatial Computing*. (Thesis, Massachusetts Institute of Technology). Retrieved from <https://acg.media.mit.edu/people/simong/thesis/SpatialComputing.pdf>
- Harley, J. M., Lajoie, S. P., Tressel, T., & Jarrell, A. (2020). Fostering positive emotions and history knowledge with location-based augmented reality and tour-guide prompts. *Learning and Instruction*, 70, 101163. doi:10.1016/j.learninstruc.2018.09.001
- Jakl, A. (2017, December 11). *Real-Time Light Estimation with Google ARCore*. Retrieved from <https://www.andreasjakl.com/real-time-light-estimation-with-google-arcore/>
- Jinyu, L., Bangbang, Y., Danpeng, C., Nan, W., Guofeng, Z., & Hujun, B. (2019). Survey and evaluation of monocular visual-inertial SLAM algorithms for augmented reality. *Virtual Reality & Intelligent Hardware*, 1(4), 386-410. doi:10.1016/j.vrih.2019.07.002
- Kán, P., & Kafumann, H. (2019). DeepLight: light source estimation for augmented reality using deep learning. *The Visual Computer*, 35(6-8), 873–883. doi:10.1007/s00371-019-01666-x
- Kowalcuk, P., Siepmann, C., & Adler, J. (2021). Cognitive, affective, and behavioral consumer responses to augmented reality in e-commerce: A comparative study. *Journal of Business Research*, 124, 357-373. doi:10.1016/j.jbusres.2020.10.050
- Liono, R. A., Amanda, N., Pratiwi, A., & Gunawan, A. A. (2021). A Systematic Literature Review: Learning with Visual by The Help of Augmented Reality Helps Students Learn Better. *Procedia Computer Science*, 179, 144-152. doi:10.1016/j.procs.2020.12.019
- Listianawati, W., Ayu, E. R., Arista, S., Stephanie, J., & Dyastuti, N. (2011, February 22). *Proses Rendering dan Animasi serta Contoh Nyatanya*. Retrieved from <https://wenythewpooh.wordpress.com/2011/02/22/proses-rendering-dan-animasi-serta-contoh-nyatanya/>
- Marques, B. A., Clua, E. W., & Vasconcelos, C. N. (2018). Deep spherical harmonics light probe estimator for mixed reality games. *Computers & Graphics*, 76, 96-106. doi:10.1016/j.cag.2018.09.003
- Masi, C. (2020, September 30). *What is the cause behind measurement noise in sensors?* Retrieved from <https://www.researchgate.net/post/What-is-the-cause-behind-measurement-noise-in-sensors>
- Mayditia, H., & Prabowo, G. S. (2008). ANALISIS PENGUKURAN KETIDAKSTABILAN BIAS DARI SENSOR GYRO FORS-4 SATELIT LAPAN-A2. *Jurnal Teknologi Dirgantara*, 6(2).
- Meenakshi, S. V., Vasudevan, S. K., Ritesh, A., & Santhosh, C. (2015). An Innovative App with for Location Finding with Augmented Reality Using CLOUD. *Procedia Computer Science*, (pp. 585-589). doi:10.1016/j.procs.2015.04.088

- Michel, T., Genevès, P., Fourati, H., & Layaïda, N. (2018). Attitude estimation for indoor navigation and augmented reality with smartphones. *Pervasive and Mobile Computing*, 46, 96-121. doi:10.1016/j.pmcj.2018.03.004
- Moezzi, R., Krcmarik, D., Hlava, J., & Cýrus, J. (2020). Hybrid SLAM modelling of autonomous robot with augmented reality device. *Materials Today: Proceedings*, (pp. 103-107). doi:10.1016/j.matpr.2020.03.036
- Nikhashemi, S., Knight, H. H., Nusair, K., & Liat, C. B. (2021). Augmented reality in smart retailing: A (n) (A) Symmetric Approach to continuous intention to use retail brands' mobile AR apps. *Journal of Retailing and Consumer Services*, 60, 102464. doi:10.1016/j.jretconser.2021.102464
- Pantuwong, N. (2016). A tangible interface for 3D character animation using augmented reality technology. *2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE)*, (pp. 1-6).
- Qin, H., Peak, D. A., & Prybutok, V. (2021). A virtual market in your pocket: How does mobile augmented reality (MAR) influence consumer decision making? *Journal of Retailing and Consumer Services*, 58, 102337. doi:10.1016/j.jretconser.2020.102337
- Roberto, R., Lima, J. P., Uchiyama, H., Teichrieb, V., & Taniguchi, R.-i. (2019). Geometrical and statistical incremental semantic modeling on mobile devices. *Computers & Graphics*, 84, 199-211. doi:10.1016/j.cag.2019.09.003
- Schechter, S. (2020, 10 20). *The Ultimate Guide to Markerless Augmented Reality*. Retrieved from <https://www.marxentlabs.com/what-is-markerless-augmented-reality-dead-reckoning/>
- Smink, A. R., Reijmersdal, E. A., Noort, G. v., & Neijens, P. C. (2020). Shopping in augmented reality: The effects of spatial presence, personalization and intrusiveness on app and brand responses. *Journal of Business Research*, 118, 474-485. doi:10.1016/j.jbusres.2020.07.018
- Suni, A. F., & Fathoni, K. (2016). Klasifikasi Shadow Algorithm. *Jurnal Teknik Elektro*, 8(2), 56-59.
- Vries, J. D. (n.d.). *Shaders*. Retrieved from <https://learnopengl.com/Getting-started/Shaders>
- Zhang, H., Zhang, J., Yin, X., Zhou, K., & Pan, Z. (2020). Cloud-to-end Rendering and Storage Management for Virtual Reality in Experimental Education. *Virtual Reality & Intelligent Hardware*, 2(4), 368-380. doi:10.1016/j.vrih.2020.07.001

## LAMPIRAN

Tabel berikut menunjukkan lampiran dari hasil kuesioner untuk mengetahui seberapa realistik hasil bayangan dari aplikasi sebagai pengukuran di bab 5, yang berisi 16 foto (8 foto menggunakan estimasi cahaya dan 8 foto tanpa menggunakan estimasi cahaya) dan dikumpulkan dari tanggal 1 Juli 2021 sampai 4 Juli 2021 oleh 52 responden.

Lampiran 1 Hasil kuesioner

No	Pertanyaan	Nilai					Total	Rata-rata
		1	2	3	4	5		
1	Berdasarkan gambar di atas, seberapa realistik intensitas cahaya dan arah bayangan karakter virtual (Total delapan foto menggunakan metode estimasi cahaya)	1	21	72	195	127	1674	4.026
2	Berdasarkan gambar di atas, seberapa realistik intensitas cahaya dan arah bayangan karakter virtual (Total delapan foto tanpa menggunakan metode estimasi cahaya)	119	183	73	25	16	884	2.125