# Babylon.js :

## *menggunakan webxr*

Agus Purwanto

October 8, 2025

.



Book Cover by ©Agus Purwanto (purwanto.agus@gmail.com)

The cover depicts a chronic overthinker trapped within their own mental complexity — a labyrinth of thoughts represented by surrounding blocks. Despite being absorbed in this self-constructed maze, there remains an awareness that the light of clarity and peace they seek lies beyond their current mental prison.

# Contents

# Part I

# Dasar

CHAPTER 1

# Perbandingan Software

Berikut adalah perbandingan beberapa software open-source untuk Augmented Reality (AR) dan Virtual Reality (VR) dari segi keunggulan, kemudahan, komunitas, popularitas, serta integrasi dengan OpenCV untuk image processing:

## 1.1 Godot Engine (Open-Source)

### 1.1.1 Keunggulan

#### 1.1.1.1 Kelebihan

Godot memiliki dukungan bawaan untuk VR dan AR yang terus berkembang. Godot sangat modular dengan arsitektur berbasis node yang memudahkan pengembang untuk mengatur berbagai elemen game, termasuk elemen VR.

#### 1.1.1.2 Lisensi

MIT License (free and open-source), sehingga Anda bebas menggunakannya tanpa batasan untuk proyek komersial maupun pribadi.

### 1.1.2 Kemudahan

— Antarmuka yang mudah dipahami untuk pemula.

— Memiliki GDScript, bahasa pemrograman scripting yang mirip dengan Python, yang membuatnya lebih ramah untuk pengguna baru.

— Dukungan untuk plugin VR di Godot cukup baik, meskipun fitur AR masih terus dalam pengembangan.

### 1.1.3 Komunitas

— Komunitas Godot tumbuh pesat dalam beberapa tahun terakhir, dengan banyak forum, Discord server, dan subreddit yang aktif.

— Tutorial dan dokumentasi yang cukup lengkap dan tersedia di berbagai platform.

### 1.1.4 Popularitas

Popularitasnya meningkat pesat, terutama dalam komunitas pengembang indie dan open-source karena fleksibilitas, lisensi terbuka, dan dukungan yang baik untuk 2D dan 3D.

### 1.1.5 Integrasi OpenCV

OpenCV dapat diintegrasikan ke dalam Godot melalui plugin atau melalui ekstensi GDNative, yang memu-

ngkinkan pengembang menggunakan pustaka eksternal seperti OpenCV untuk pemrosesan gambar.

## 1.2 Babylon.js (Open-Source)

### 1.2.1 Keunggulan

#### 1.2.1.1 Kelebihan

Babylon.js mendukung **WebXR** secara penuh, yang memungkinkan Anda membuat pengalaman VR dan AR yang berjalan langsung di browser. Ini membuatnya ideal untuk aplikasi yang bisa diakses lintas platform tanpa harus menginstal aplikasi tambahan.

#### 1.2.1.2 Fitur kuat untuk 3D dan grafis

Babylon.js mendukung rendering grafis 3D yang canggih dan memiliki banyak fitur bawaan seperti

— physics engine,

— lighting,

— shadows,

yang membuatnya unggul untuk aplikasi dengan kebutuhan grafis tinggi.

#### 1.2.1.3 Lisensi

Apache License 2.0 (sangat terbuka dan fleksibel untuk proyek komersial maupun pribadi).

### 1.2.2 Kemudahan

— Babylon.js relatif mudah digunakan oleh pengembang yang familiar dengan JavaScript dan web development. Sintaksnya berbasis JavaScript dan bekerja dengan baik bersama teknologi web lainnya seperti HTML dan CSS.

— Mendukung plugin dan ekosistem yang memungkinkan Anda dengan cepat membangun dan mengimplementasikan aplikasi AR/VR yang kaya fitur.

— Komponen **Inspector** memungkinkan debugging visual yang sangat mudah.

### 1.2.3 Komunitas

— Komunitas Babylon.js sangat aktif dan terus berkembang. Terdapat forum diskusi resmi, grup Discord, serta banyak dokumentasi dan tutorial online yang mempermudah pengguna baru dalam memulai.

— Dokumentasi resmi Babylon.js sangat komprehensif dan mencakup banyak tutorial serta contoh-contoh kode, termasuk topik WebXR.

### 1.2.4 Popularitas

— Babylon.js sangat populer di kalangan pengembang web yang ingin membuat aplikasi 3D yang bisa diakses melalui browser, terutama untuk

pengembangan game, simulasi, dan aplikasi berbasis visualisasi 3D.

— Seiring dengan perkembangan WebXR, Babylon.js menjadi salah satu framework paling populer untuk pengembangan AR dan VR berbasis web.

### 1.2.5 Integrasi OpenCV

— OpenCV.js (pustaka JavaScript dari OpenCV) dapat digunakan dalam proyek Babylon.js, memungkinkan pemrosesan gambar dalam konteks 3D yang dikembangkan dengan Babylon.js.

— Pengembang dapat menggabungkan OpenCV untuk tugas-tugas pemrosesan gambar seperti deteksi objek, pengenalan wajah, dan lainnya dalam aplikasi AR/VR yang dibuat dengan Babylon.js.

Contoh Penggunaan Babylon.js dalam AR/VR:

**AR:** Babylon.js mendukung **WebXR AR** yang memungkinkan Anda membangun aplikasi berbasis AR yang dapat diakses melalui perangkat mobile atau desktop dengan browser yang mendukung WebXR. Misalnya, Anda bisa membuat aplikasi yang menampilkan objek 3D di dunia nyata menggunakan kamera perangkat pengguna.

**VR:** Babylon.js memungkinkan pengembangan **VR** penuh dengan dukungan untuk perangkat seperti Oculus, HTC Vive, dan perangkat lain yang mendukung **WebXR VR**. Pengalaman VR bisa dibuat

dengan mudah menggunakan API WebXR Babylon.js untuk navigasi, interaksi, dan rendering dunia virtual.

Babylon.js sangat menarik untuk dibandingkan, khususnya di dalam ekosistem pengembangan web. Dalam konteks AR/VR, Babylon.js menonjol karena:

**Kemudahan Penggunaan:** Pengembangan AR/VR dengan Babylon.js sangat sederhana bagi pengembang JavaScript.

**Kuat di Sisi WebXR:** Mendukung AR dan VR langsung di browser tanpa perlu aplikasi tambahan, yang merupakan keunggulan utama dalam hal aksesibilitas.

**Komunitas dan Dokumentasi:** Sangat kuat dengan dokumentasi yang luas dan komunitas yang aktif, menjadikannya pilihan populer untuk pengembangan aplikasi berbasis 3D dan interaktif.

## 1.3 AR.js (Open-Source)

### 1.3.1 Keunggulan

#### 1.3.1.1 Kelebihan

AR.js merupakan pustaka AR berbasis web yang dapat digunakan untuk mengembangkan aplikasi AR tanpa perlu menggunakan aplikasi mobile native. Ini menjadikannya sangat fleksibel untuk aplikasi web berbasis AR.

#### 1.3.1.2 Lisensi

Lisensi MIT yang sangat terbuka.

### 1.3.2 Kemudahan

— Mudah digunakan untuk pengembang yang familiar dengan pengembangan web, terutama menggunakan HTML, JavaScript, dan frameworks web seperti Three.js.

— Tidak perlu pengetahuan khusus tentang VR/AR SDK yang rumit.

### 1.3.3 Komunitas

— Komunitas ini terhubung erat dengan komunitas pengembang web, terutama mereka yang menggunakan Three.js.

— Dokumentasi relatif cukup, tetapi mungkin tidak sekomprehensif proyek besar lainnya.

### 1.3.4 Popularitas

— Popular dalam kalangan pengembang web karena integrasinya yang mudah dengan teknologi web yang sudah ada.

— Banyak digunakan untuk aplikasi AR sederhana di web.

### 1.3.5 Integrasi OpenCV

OpenCV dapat diintegrasikan ke AR.js dengan memanfaatkan pustaka JavaScript OpenCV.js. Hal ini memungkinkan penggunaan fungsi pemrosesan gambar langsung dalam proyek AR berbasis web.

## 1.4 A-Frame (Open-Source)

### 1.4.1 Keunggulan

#### 1.4.1.1 Kelebihan

— A-Frame adalah framework VR berbasis web yang dikembangkan di atas Three.js dan sangat mudah digunakan untuk membuat pengalaman VR yang ringan di browser.

— Mendukung WebXR untuk integrasi AR dan VR dengan API berbasis web.

### 1.4.2 Kemudahan

— A-Frame menggunakan HTML dan JavaScript yang memungkinkan pengembang web membuat pengalaman VR dengan cepat tanpa mempelajari SDK VR yang kompleks.

— Sintaks berbasis deklaratif (HTML) memudahkan pemula dalam pembuatan dunia 3D dan VR.

### 1.4.3 Komunitas

— A-Frame memiliki komunitas yang aktif, terutama dalam ekosistem pengembangan web. -

— Banyak tutorial, dokumentasi, dan proyek yang dibagikan oleh komunitas.

### 1.4.4 Popularitas

— Salah satu framework VR berbasis web yang paling populer dan banyak digunakan.

— Digunakan oleh banyak proyek untuk menciptakan pengalaman VR/AR ringan yang dapat diakses langsung melalui browser.

### 1.4.5 Integrasi OpenCV

A-Frame dapat memanfaatkan OpenCV melalui OpenCV.js atau pustaka JavaScript yang dapat digunakan dalam browser. Hal ini memudahkan integrasi pemrosesan gambar langsung di aplikasi VR berbasis web.

## 1.5 OpenXR (Open-Source)

### 1.5.1 Keunggulan

#### 1.5.1.1 Kelebihan

OpenXR adalah standar terbuka untuk mengembangkan pengalaman VR dan AR di berbagai perangkat. Dibuat oleh Khronos Group, OpenXR menawarkan API yang

seragam untuk berinteraksi dengan perangkat VR/AR seperti Oculus, HTC Vive, dan Hololens.

#### 1.5.1.2 Lisensi

Lisensi terbuka yang menjamin interoperabilitas antar perangkat.

### 1.5.2 Kemudahan

API OpenXR cukup mendalam dan rumit untuk dipelajari dibandingkan framework lain seperti A-Frame atau AR.js, terutama jika Anda belum familiar dengan pengembangan VR/AR di tingkat rendah.

### 1.5.3 Komunitas

— Komunitas pengembang OpenXR banyak terdiri dari pengembang perangkat keras dan perangkat lunak besar yang fokus pada ekosistem VR/AR tingkat lanjut.

— Komunitas OpenXR juga terdiri dari pengembang perangkat independen.

### 1.5.4 Popularitas

Meskipun kurang populer di kalangan pengembang indie, OpenXR sangat penting dalam standar industri VR/AR, terutama untuk perusahaan besar yang ingin membuat aplikasi lintas perangkat.

### 1.5.5 Integrasi OpenCV

OpenXR sendiri tidak memiliki integrasi bawaan dengan OpenCV, tetapi dapat diintegrasikan dengan aplikasi yang menggunakan OpenCV untuk pemrosesan gambar melalui teknik standar seperti C++ bindings atau API eksternal.

## 1.6 Mozilla Hubs (Open-Source)

### 1.6.1 Keunggulan

#### 1.6.1.1 Kelebihan

Mozilla Hubs adalah platform sosial VR berbasis web yang memungkinkan pengguna untuk membuat ruang virtual dan berinteraksi dengan orang lain di dunia virtual. Semua ini dapat diakses langsung dari browser tanpa perangkat lunak khusus.

#### 1.6.1.2 Lisensi

GPLv3, yang membuatnya menjadi open-source sepenuhnya.

### 1.6.2 Kemudahan

— Sangat mudah digunakan bagi pengguna biasa, karena hanya membutuhkan browser untuk mengakses ruang VR.

— Pengembangan konten di Hubs dapat dilakukan melalui antarmuka yang sederhana.

### 1.6.3 Komunitas

Komunitas penggunanya cukup aktif, terutama di kalangan pendidik dan kreator yang ingin membuat ruang VR sosial untuk interaksi virtual.

### 1.6.4 Popularitas

— Cukup populer di kalangan komunitas open-source VR dan pengguna yang mencari solusi mudah untuk menciptakan ruang VR sosial.

— Digunakan terutama untuk pertemuan dan konferensi virtual.

### 1.6.5 Integrasi OpenCV

Tidak ada integrasi bawaan untuk OpenCV, namun, jika pengembang ingin melakukan pemrosesan gambar dalam konteks Mozilla Hubs, OpenCV dapat digunakan secara terpisah di backend untuk memberikan masukan ke dalam dunia Hubs.

## 1.7 Processing

Perbandingan antara `Processing` dan `Babylon.js` dalam konteks `WebXR` bergantung pada kebutuhan spesifik Anda, karena kedua framework ini memiliki pendekatan yang berbeda.

### 1.7.1 Mengenai Babylon.js

**Contoh kode Babylon.js untuk WebXR:**

**Program 1.1: Script untuk** `Babylon.js`

```
const createScene = function () {
    const scene = new BABYLON.Scene(
        engine);
    const camera = new BABYLON.
        ArcRotateCamera("camera", Math
        .PI / 2, Math.PI / 2, 2,
        BABYLON.Vector3.Zero(), scene)
        ;
    camera.attachControl(canvas, true)
        ;

    const light = new BABYLON.
        HemisphericLight("light", new
        BABYLON.Vector3(1, 1, 0),
        scene);

    // Create an XR experience
    const xr = scene.
        createDefaultXRExperienceAsync
        ({});

    // Create a simple sphere
```

```
12          const sphere = BABYLON.MeshBuilder
                .CreateSphere("sphere", {
                diameter: 1 }, scene);
13          return scene;
14      };
```

#### 1.7.1.1 Spesialisasi

Babylon.js adalah framework yang dirancang khusus untuk 3D rendering di web dan mendukung WebXR secara bawaan. Ini berarti Babylon.js sangat kuat untuk proyek WebXR (Virtual Reality dan Augmented Reality) langsung di browser tanpa memerlukan plugin tambahan.

#### 1.7.1.2 Keunggulan

**WebXR Native Support:** Babylon.js menyediakan integrasi langsung untuk WebXR, sehingga memudahkan Anda untuk membuat pengalaman VR/AR di browser dengan minimal setup.

**Pustaka Lengkap:** Babylon.js menawarkan pustaka yang sangat lengkap untuk pengelolaan objek 3D, lighting, animasi, dan fisika, yang ideal untuk membuat dunia virtual yang interaktif.

**Komunitas dan Ekosistem Besar:** Babylon.js memiliki komunitas besar dengan dokumentasi yang bagus, serta banyak contoh penggunaan WebXR.

**Kinerja:** Kinerja tinggi untuk rendering 3D di browser dengan dukungan WebGL, yang membuatnya cocok untuk aplikasi WebXR yang memerlukan grafis yang halus dan interaktif.

### 1.7.2 Mengenai Processing

#### 1.7.2.1 Spesialisasi

Processing adalah alat dan bahasa pemrograman yang lebih berfokus pada seni visual, animasi, dan kreativitas digital. Ini awalnya dirancang untuk artis dan desainer yang ingin membuat visualisasi dengan cepat dan mudah. Processing memiliki versi berbasis JavaScript yang disebut `p5.js`, yang bisa digunakan untuk aplikasi web, tetapi tidak didesain secara khusus untuk WebXR atau rendering 3D yang kompleks.

#### 1.7.2.2 Keunggulan

**Sederhana dan Mudah Dipelajari:** Processing dan p5.js menawarkan antarmuka pemrograman yang lebih sederhana, cocok untuk memulai atau membuat eksperimen visual dan interaksi sederhana.

**Fokus pada Kesenian dan Interaksi 2D/3D Sederhana:** Lebih cocok untuk prototyping visual, seni generatif, dan eksplorasi estetika, tetapi kurang bertenaga untuk rendering 3D yang kompleks dan pengalaman XR yang canggih.

**Fleksibilitas dalam 2D:** Jika fokus Anda lebih pada pembuatan konten interaktif yang ringan atau

berbasis 2D, Processing (p5.js) bisa menjadi pilihan yang lebih mudah.

Namun, untuk `WebXR`, `Processing/p5.js` tidak memiliki dukungan langsung. Anda akan memerlukan banyak tambahan atau menggunakan pustaka eksternal untuk mencapai tujuan yang sama, membuatnya kurang praktis dibandingkan dengan Babylon.js.

### 1.7.3 Mana yang Lebih Baik untuk WebXR?

`Babylon.js` lebih unggul jika fokus Anda adalah membuat konten 3D interaktif yang rumit dan rendering yang mulus di browser, terutama dalam konteks WebXR. Babylon.js secara langsung mendukung WebXR, dan Anda dapat memanfaatkan alat-alat canggih untuk mengembangkan pengalaman XR yang lebih kompleks dengan efisiensi tinggi.

`Processing/p5.js` lebih baik jika Anda berfokus pada eksperimen visual sederhana atau seni digital, tetapi ini tidak ideal untuk proyek WebXR yang membutuhkan 3D rendering dan interaktivitas tingkat lanjut.

### 1.7.4 Kesimpulan

Jika tujuan utama Anda adalah membuat konten `WebXR` interaktif, `Babylon.js` adalah pilihan yang jauh lebih baik dan efisien. Namun, jika Anda menginginkan platform yang lebih sederhana untuk eksplorasi vi-

sual atau seni, `Processing/p5.js` bisa menjadi alternatif, tetapi akan membutuhkan lebih banyak upaya untuk membuat pengalaman XR yang sebanding dengan yang bisa dicapai dengan Babylon.js.

## 1.8 Kesimpulan

**Godot Engine** menonjol dalam hal fleksibilitas dan kemudahan, serta komunitas yang kuat. Sangat cocok untuk pengembang yang ingin membuat aplikasi VR dan AR secara terpisah atau dalam game. Integrasi OpenCV dimungkinkan melalui GDNative.

**Babylon.js** ideal untuk aplikasi berbasis web. Godot Engine lebih cocok untuk game dan aplikasi berbasis desktop. AR.js dan A-Frame fokus pada pengembangan berbasis web juga, tetapi Babylon.js lebih unggul dalam hal fitur grafis 3D yang lebih canggih. Jika Anda ingin menggabungkan WebXR dan kemampuan OpenCV untuk pemrosesan gambar, Babylon.js sangat fleksibel karena dapat mengintegrasikan OpenCV.js.

**AR.js dan A-Frame** kuat dalam hal pengembangan berbasis web untuk AR dan VR, dengan dukungan yang baik untuk WebXR dan dapat diakses langsung melalui browser. Integrasi OpenCV dapat dilakukan melalui OpenCV.js.

**OpenXR** adalah standar untuk perangkat VR/AR dan menyediakan API tingkat rendah, cocok untuk

proyek lintas platform yang membutuhkan dukungan perangkat keras luas.

**Mozilla Hubs** lebih fokus pada interaksi sosial berbasis web dan mudah diakses oleh pengguna biasa, tetapi tidak memiliki integrasi langsung dengan OpenCV.

Untuk OpenCV, Godot, Babylon.js, AR.js, dan A-Frame menawarkan integrasi yang paling mudah, terutama jika digunakan melalui versi JavaScript atau binding eksternal.

CHAPTER 2

# Web Developer Tool

Web developer tools yang tersedia di Google Chrome (Chrome DevTools) dan Mozilla Firefox (Firefox Developer Tools) dapat digunakan sebagai alat bantu pengembangan web, meskipun mereka tidak sepenuhnya menggantikan Integrated Development Environment (IDE) seperti Visual Studio Code, Atom, atau Sublime Text. Namun, mereka memiliki fitur-fitur yang sangat berguna untuk debugging, pengujian, dan pengembangan web secara langsung di browser.

Berikut adalah panduan komprehensif tentang penggunaan Chrome DevTools dan Firefox Developer Tools sebagai alat pengembangan, serta perbandingannya dengan IDE tradisional.

## 2.1 Panduan Menggunakan Chrome DevTools dan Firefox Developer Tools sebagai Alat Pengembangan

### 2.1.1 Chrome DevTools

Chrome DevTools adalah seperangkat alat yang disematkan di Google Chrome untuk membantu pengembang web dalam mengembangkan, menguji, dan mendebug aplikasi web.

Fitur Utama:

**Elements Panel:** Memeriksa dan memodifikasi HTML dan CSS secara real-time.

**Console:** Menjalankan JavaScript, menampilkan log, dan men-debug kode.

**Sources Panel:** Men-debug JavaScript, menambahkan breakpoint, dan mengedit file sumber.

**Network Panel:** Memantau permintaan jaringan, menganalisis kinerja, dan mengoptimalkan beban halaman.

**Performance Panel:** Menganalisis kinerja aplikasi dan mengidentifikasi bottleneck.

**Application Panel:** Mengelola penyimpanan lokal, cache, dan database.

**Lighthouse:** Alat untuk mengaudit performa, aksesibilitas, SEO, dan lainnya.

Cara Menggunakan:

— Buka Chrome dan tekan 'F12' atau 'Ctrl + Shift + I' (Windows/Linux) atau 'Cmd + Option + I' (Mac) untuk membuka DevTools.

— Gunakan `Elements Panel` untuk mengedit HTML/CSS secara langsung.

— Gunakan `Console` untuk mengetes kode JavaScript.

— Gunakan `Sources Panel` untuk men-debug kode JavaScript.

— Gunakan `Network Panel` untuk memantau permintaan HTTP/HTTPS.

### 2.1.2 Firefox Developer Tools

Firefox Developer Tools adalah alat serupa yang disediakan oleh Mozilla Firefox untuk pengembangan web.

Fitur Utama:

**Inspector:** Memeriksa dan memodifikasi HTML dan CSS.

**Console:** Menjalankan JavaScript dan men-debug kode.

**Debugger:** Men-debug JavaScript dengan breakpoint dan call stack.

**Network Monitor:** Memantau permintaan jaringan.

**Performance Tool:** Menganalisis kinerja aplikasi.

**Storage Panel:** Mengelola penyimpanan lokal, session, dan cookie.

**Accessibility Panel:** Memeriksa aksesibilitas halaman web.

Cara Menggunakan:

— Buka Firefox dan tekan 'F12' atau 'Ctrl + Shift + I' (Windows/Linux) atau 'Cmd + Option + I' (Mac) untuk membuka Developer Tools.

— Gunakan `Inspector` untuk mengedit HTML/CSS.

— Gunakan `Console` untuk mengetes JavaScript.

— Gunakan `Debugger` untuk men-debug kode JavaScript.

— Gunakan `Network Monitor` untuk memantau permintaan jaringan.

## 2.2 Perbandingan dengan IDE Lain (Visual Studio Code, Atom, Sublime Text)

| Kemampuan | Chrome DevTools | Firefox DevTools | Visual Studio Code | Atom | Sublime Text |
|---|---|---|---|---|---|
| Editing HTML/CSS | Ya (real-time) | Ya (real-time) | Ya | Ya | Ya |
| Debugging JavaScript | Ya | Ya | Ya (dengan ekstensi) | Ya (dengan ekstensi) | Ya (dengan ekstensi) |
| Version Control | Tidak | Tidak | Ya (Git terintegrasi) | Ya (Git terintegrasi) | Dengan plugin |
| Terminal Integrasi | Tidak | Tidak | Ya | Ya | Dengan plugin |
| Ekstensi/Plugin | Terbatas | Terbatas | Sangat banyak | Banyak | Banyak |
| Performance Analysis | Ya | Ya | Dengan ekstensi | Dengan ekstensi | Dengan ekstensi |
| Cross-Platform | Ya (hanya Chrome) | Ya (hanya Firefox) | Ya | Ya | Ya |
| Harga | Gratis | Gratis | Gratis | Gratis | Berbayar (dengan uji coba gratis) |

## 2.3 Kapan Menggunakan Chrome/-Firefox DevTools vs IDE

### 2.3.1 Gunakan Chrome/Firefox DevTools

— Untuk debugging dan pengujian langsung di browser.

— Untuk memeriksa dan memodifikasi HTML/CSS secara real-time.

— Untuk menganalisis kinerja jaringan dan aplikasi.

— Untuk pengujian aksesibilitas dan SEO.

### 2.3.2 Gunakan IDE (Visual Studio Code, Atom, Sublime)

— Untuk menulis kode dalam skala besar.

— Untuk manajemen proyek dan version control (Git).

— Untuk pengembangan aplikasi full-stack (back-end + frontend).

— Untuk menggunakan ekstensi dan alat otomatisasi (linting, testing, dll).

## 2.4 Kesimpulan

Chrome DevTools dan Firefox Developer Tools adalah alat yang sangat kuat untuk pengembangan web, terutama untuk debugging dan pengujian langsung di browser.

**Namun, mereka tidak menggantikan IDE seperti Visual Studio Code, Atom, atau Sublime Text untuk pengembangan proyek skala besar dan manajemen kode yang lebih kompleks. Kombinasi keduanya (menggunakan IDE untuk menulis kode dan DevTools untuk debugging) adalah pendekatan terbaik untuk pengembangan web modern.**

CHAPTER 3

# IDE Babylonjs

**Berikut penjelasan perbedaan antara** `Babylon.js Editor` **(**https://editor.babylonjs.com/**),** `NME (Node Material Editor)` **(**https://nme.babylonjs.com/**), dan** `NGE (Node Geometry Editor)` **(**https://nge.babylonjs.com/**), serta hubungannya.**

## 3.1 Babylon.js Editor

### 3.1.1 Fungsi

**Sebuah editor visual lengkap untuk membuat aplikasi 3D/web dengan Babylon.js.**

- **Mendukung pembuatan scene, material, animasi, scripting (TypeScript/JavaScript), dan ekspor ke berbagai format.**
- **Cocok untuk pengembangan proyek end-to-end (game, aplikasi web 3D, dll).**

### 3.1.2 Fitur Utama

- **Integrasi dengan semua fitur Babylon.js.**
- **Antarmuka mirip Unity/Blender (hierarki objek, inspector, dll).**

## 3.2 NME (Node Material Editor)

### 3.2.1 Fungsi

Khusus untuk membuat material shader menggunakan sistem node-based.

— Analog dengan Shader Graph di Unity atau Blender's Shader Editor.

— Hasilnya bisa dipakai di Babylon.js Editor atau proyek standalone.

### 3.2.2 Contoh Penggunaan

— Membuat material air

— logam atau

— efek visual kompleks tanpa menulis kode GLSL/HLSL manual

## 3.3 NGE (Node Geometry Editor)

### 3.3.1 Fungsi

Khusus untuk pemodelan geometri procedural menggunakan node.

— Analog dengan Geometry Nodes di Blender.

— Menghasilkan mesh secara dinamis berdasarkan parameter.

### 3.3.2 Contoh Penggunaan

— Membangun gedung

— vegetasi

— atau objek procedural lainnya.

## 3.4 Hubungan Ketiganya

### 3.4.1 NME + NGE $\neq$ Babylon.js Editor

— NME dan NGE adalah alat spesifik (material/-geometry), sedangkan Babylon.js Editor adalah lingkungan pengembangan terintegrasi.

— Hasil dari NME/NGE bisa diimpor ke Babylon.js Editor, tetapi Babylon.js Editor memiliki fitur lebih luas (scene management, scripting, dll).

### 3.4.2 Diagram Alur

— NGE (Geometry) $\rightarrow$ Ekspor Mesh $\rightarrow$ Babylon.js Editor (Integrasi Scene)

— NME (Material) $\rightarrow$ Ekspor Shader $\rightarrow$ Babylon.js Editor (Penerapan Material)

## 3.5 Kapan Menggunakan yang Mana?

| Alat | Gunakan Untuk | Contoh Kasus |
|---|---|---|
| Babylon.js Editor | Proyek lengkap (scene, interaksi, animasi) | Game, Aplikasi Web 3D |
| NME | Material/shader kompleks | Air, Api, Efek Kustom |
| NGE | Pemodelan procedural | Kota, Hutan, Objek Teracak |

## 3.6 Catatan Integrasi

### 3.6.1 Ekspor/Impor

— Hasil NME/NGE bisa disimpan sebagai '.json' atau kode JavaScript, lalu di-load di Babylon.js Editor.

— Di Babylon.js Editor, Anda bisa membuka NME/NGE secara terintegrasi via tombol Edit Material/Geometry.

### 3.6.2 Versi Standalone

NME/NGE juga tersedia sebagai komponen terpisah di repositori Babylon.js jika ingin dipakai di proyek custom.

## 3.7 Kesimpulan

— Babylon.js Editor adalah "IDE" utama.

— NME dan NGE adalah alat pendukung untuk tugas spesifik (material/geometry).

— Anda bisa menggunakan ketiganya secara terpisah atau dikombinasikan tergantung kebutuhan.

Jika tujuan Anda adalah membuat aplikasi 3D penuh, mulailah dengan Babylon.js Editor, lalu manfaatkan NME/NGE saat diperlukan

CHAPTER 4

# Dasar Pemrograman

## 4.1 Pendahuluan

Buku ini dibuat sebagai bagian dari perkuliahan teknologi dan media pembelajaran. Dalam perkuliahan tersebut, salah satu materinya adalah pembuatan animasi, game atau xr dimana x=a (augmented) atau v (virtual). Dalam materi ini digunakan babylonjs (https://www.babylonjs.com/) karena keunggulannya seperti dibahas pada subbab 1.2.1 pada Halaman 5.

Program berbahasa javascript, dapat dijalankan melalui browser melalui file html. Program yang dibahas disini juga ditampilkan di https://youtube.com/shorts/fUWVTT1D80o?si=-OxLIVJEmApVEYbm. Versi geometri simpul (*geometry node*) yang merupakan cara pemrograman secara visual (bukan script) ditampilkan di https://youtube.com/shorts/qJaN89BUGd0?si=lkaSwQKifJfhZ3t Cara visual tidak dibahas disini karena kurang berlaku umum, walaupun lebih menarik untuk anak-anak yang baru mulai belajar pemrograman komputer.

## 4.2 File html

**Dalam pembahasan ini, file javascrit dipanggil oleh file html (lihat baris ke 9 di bawah ini). Sebelum pemanggilan tersebut, pendefinisian canvas perlu dilakukan seperti pada baris ke 8.**

**Program 4.1: file.html**

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="
         width=device-width,
         initial-scale=1.0">
6    <title>Babylon.js MWE - Kalimat
         Bersyarat</title>
7    <script src="https://cdn.babylonjs.com
         /babylon.js"></script>
8    <canvas id="renderCanvas" style="width
         : 100%; height: 100%;"></canvas>
9    <script type="text/javascript" src="2.
         js"></script>
10 </head>
11 <body>
12 Ini adalah contoh kalimat bersyarat
       menggunakan if.  Dalam kasus ini
       index
```

```
13  (dimulai dengan nol) genap diberi warna
        merah.  Lainnya biru.
14  </body>
15  </html>
```

## 4.3 File JavaScript

### 4.3.1 Kerangka Umum

Program javascript yang dipanggil pada Script di Sub-bab 4.2 Halaman 36 dapat dilihat pada listing di bawah ini. Kerangka umumnya, sebelum objek mesh masuk, dapat dilihat di bawah ini.

Program 4.2: 2.js Kerangka (selalu ada)

```
1  // Get the canvas element
2  var canvas = document.getElementById('
       renderCanvas');
3
4  // Create BabylonJS engine
5  var engine = new BABYLON.Engine(canvas,
       true);
6
7  // Create a basic scene
8  var createScene = function () {
9    var scene = new BABYLON.Scene(engine);
10
```

```
11    // Add a camera and position it
12    var camera = new BABYLON.
          ArcRotateCamera("camera",
13      Math.PI / 2, Math.PI / 2, 4,
14      new BABYLON.Vector3(0, 0, 0), scene)
            ;
15    camera.attachControl(canvas, true);
16
17    // Add a light
18    var light = new BABYLON.PointLight("
          pointLight",
19      new BABYLON.Vector3(1, 1, 1), scene)
            ;
20    light.diffuse = new BABYLON.Color3(1,
          1, 1);  // white light
21
22    return scene;
23  };
24
25  // Call the createScene function
26  var scene = createScene();
27
28  // Render loop
29  engine.runRenderLoop(function () {
30    scene.render();
31  });
32
```

```
33 // Resize the engine if the window is
       resized
34 window.addEventListener('resize',
       function () {
35   engine.resize();
36 });
```

Perhatikan bahwa canvas yang sudah didefinisikan bernama renderCanvas di file html, dipanggil pada baris ke 2. Scene, camera dan light adalah standard tampilan 3D. Paling tidak, kita pelajari tiga hal penting dari listing program js di bawah ini:

1. Library. Pernyataan yang diawali dengan `BABYLON` didefinisikan dalam library/pustaka `https://cdn.babylonjs.com/babylon.js` yang dipanggil di baris ke 7 file html. Untuk pernyataan lain, pustaka Application Programming Interface (API) dapat dilihat di `https://doc.babylonjs.com/typedoc/modules/BABYLON`.

2. Variable. Huruf yang diawali dengan `let` adalah deklarasi bahwa akan muncul variable terkait dengan nilai yang diberikan di awal tersebut. Huruf yang diawali dengan `const` mewakili konstanta yang nilainya diberikan saat pertama dipanggil dan tidak bisa diubah setelah pemberian nilai awal tersebut.

## 4.4 JavaScript utuh

Menyambung kerangka pada Subbab 4.3.1 pada Halaman 37, berikut adalah penjelasan tambahan untuk program javascript utuh:

1. Kalimat bersyarat menggunakan if (baris ke 32) dengan jika tidaknya (*else*, baris ke 34). Kalimat if ini memastikan jika i merupakan bulat, maka bolanya berwarna merah. Jika i tidak bulat (=ganjil), maka bola berwarna biru.

2. Kalimat berulang menggunakan for (baris ke 26). Tata bahasa for seperti ini juga dipakai di bahasa pemrograman lain, misalnya di C, C++ dan golang. Ada 3 komponen yang ada pada for ini: nilai awal (let i=0), syarat berhenti (i<2), dan modifikasi nilai (i++).

Program 4.3: 2.js Utuh

```
1  // Get the canvas element
2  const canvas = document.getElementById("
       renderCanvas");
3
4  // Create the Babylon.js engine
5  const engine = new BABYLON.Engine(canvas
       , true);
6
7  // Create the scene
8  // var createScene = function () {
```

```
9  const createScene = () => {
10   // Create a basic scene with a camera
         and a light
11   const scene = new BABYLON.Scene(engine
         );
12   const camera = new BABYLON.
         ArcRotateCamera("camera", Math.PI
         / 2,
13     Math.PI / 4, 10, BABYLON.Vector3.
           Zero(), scene);
14   camera.attachControl(canvas, true);
15
16   const light = new BABYLON.
         HemisphericLight("light",
17     new BABYLON.Vector3(1, 1, 0), scene)
           ;
18
19   // Material to change the colors of
         the spheres
20   const redMaterial = new BABYLON.
         StandardMaterial("redMat", scene);
21   redMaterial.diffuseColor = new BABYLON
         .Color3(1, 1, 0);
22
23   const blueMaterial = new BABYLON.
         StandardMaterial("blueMat", scene)
         ;
24   blueMaterial.diffuseColor = new
```

```
        BABYLON.Color3(0, 0, 1);
25
26    // Create a loop to generate multiple
          spheres
27    for (let i = 0; i < 2; i++) {
28      const sphere = BABYLON.MeshBuilder.
            CreateSphere(`sphere${i}`,
29        { diameter: 1 }, scene);
30      sphere.position.x = i * 2; //
            Positioning spheres on the X-
            axis
31
32      // Conditional statement to assign
            material
33      if (i % 2 === 0) {
34        sphere.material = redMaterial;  //
              Red for even indices
35      } else {
36        sphere.material = blueMaterial; //
              Blue for odd indices
37      }
38    }
39    return scene;
40  };
41
42  // Call the createScene function to set
        up the scene
43  const scene = createScene();
```

```
44
45 // Render loop
46 engine.runRenderLoop(() => {
47   scene.render();
48 });
49
50 // Resize event handler
51 window.addEventListener("resize", () =>
       {
52   engine.resize();
53 });
```

# Part II

# Animasi

CHAPTER 5

# Merge

Seringkali, objek terdiri dari lebih dari satu mesh. Saat di pindahkan, sering pula keseluruhan mesh ikut pindah. Dalam hal seperti ini, maka semua mesh terkait bisa digabungkan

## 5.1 Statis

### 5.1.1 File html

Program 5.1: index.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="
          width=device-width,
          initial-scale=1.0">
6     <title>BabylonJS MWE</title>
7     <script src="https://cdn.babylonjs.
          com/babylon.js"></script>
8     <script src="app.js" defer></script>
```

```
</head>
<body>
    <canvas id="renderCanvas" style="
        width: 100%; height: 100%;"><
        /canvas>
</body>
</html>
```

### 5.1.2 File JavaScript

**Program 5.2: app.js**

```
// Get the canvas element
var canvas = document.getElementById('
    renderCanvas');

// Create BabylonJS engine
var engine = new BABYLON.Engine(canvas,
    true);

var createScene = function () {

  // This creates a basic Babylon Scene
      object (non-mesh)
  var scene = new BABYLON.Scene(engine);

  // This creates and positions a free
      camera (non-mesh)
```

```
13  var camera = new BABYLON.FreeCamera("
        camera1",
14    new BABYLON.Vector3(0, 5, -10),
          scene);
15  // This targets the camera to scene
        origin
16  camera.setTarget(BABYLON.Vector3.Zero
        ());
17  // This attaches the camera to the
        canvas
18  camera.attachControl(canvas, true);
19
20  // This creates a light, aiming 0,1,0
        - to the sky (non-mesh)
21  var light = new BABYLON.
        HemisphericLight("light1",
22    new BABYLON.Vector3(0, 1, 0), scene)
          ;
23  // Default intensity is 1. Let's dim
        the light a small amount
24  light.intensity = 0.7;
25
26  // Materials
27  var mat0 = new BABYLON.
        StandardMaterial('mat0', scene);
28  mat0.diffuseColor = new BABYLON.Color3
        (1, 1, 0);
29
```

```
30  var mat1 = new BABYLON.
        StandardMaterial('mat1', scene);
31  mat1.diffuseColor = new BABYLON.Color3
        (1, 0, 0);
32
33  var mat2 = new BABYLON.
        StandardMaterial('mat2', scene);
34  mat2.diffuseColor = new BABYLON.Color3
        (0, 1, 0);
35
36  //Our built-in 'ground' shape. Params:
         name, width, depth,
37  //subdivs, scene
38  var ground = BABYLON.Mesh.CreateGround
        ("ground1", 6, 6, 2, scene);
39  ground.material = mat0
40
41  var sphere = BABYLON.MeshBuilder.
        CreateSphere("sphere1",
42    {diameter: 2, segments: 16}, scene);
43  sphere.material = mat1;
44  sphere.position.y = 1;
45
46  var cube = BABYLON.MeshBuilder.
        CreateBox("cube",
47    { size: 1, height: 3 }, scene);
48  cube.position = new BABYLON.Vector3(1,
         1.5, 0);
```

```
49    cube.material = mat2;
50
51    //parameters - arrayOfMeshes,
          disposeSource, allow32BitsIndices,
52    //meshSubclass, subdivideWithSubMeshes
          , multiMultiMaterial
53    var mesh = BABYLON.Mesh.MergeMeshes([
          sphere, cube],
54      true, true, undefined, false, true);
55    mesh.position = new BABYLON.Vector3(4,
           0, 0);
56
57    return scene;
58  };
59
60  //Call the createScene function
61  var scene = createScene();
62
63  //Render loop
64  engine.runRenderLoop(function () {
65    scene.render();
66  });
67
68  //Resize the engine if the window is
      resized
69  window.addEventListener('resize',
      function () {
70    engine.resize();
```

```
71 });
```

**Latihan 1: Cobalah**

Jalankan program di atas dengan klik index.html. Hilangkan baris ke 55 di file app.js. Lalu jalankan kembali dan perhatikan perbedaannya.

## 5.2 Animasi

Berikut adalah program serupa, namun gabungan mesh terkait bergerak ke atas saat halaman browser dimasuki (di render).

### 5.2.1 File html

Program 5.3: index.html

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="
          width=device-width,
          initial-scale=1.0">
6     <title>BabylonJS MWE</title>
7     <script src="https://cdn.babylonjs.
          com/babylon.js"></script>
8     <script src="app.js" defer></script>
```

```
 9  </head>
10  <body>
11      <canvas id="renderCanvas" style="
            width: 100%; height: 100%;"><
            /canvas>
12  </body>
13  </html>
```

### 5.2.2 File JavaScript

**Penambahan dibandingkan dengan listing yang di bahas di subsection 5.1.2 pada baris 63-75.**

**Program 5.4: app.js**

```
 1  // Get the canvas element
 2  var canvas = document.getElementById('
        renderCanvas');
 3
 4  // Create BabylonJS engine
 5  var engine = new BABYLON.Engine(canvas,
        true);
 6
 7  var createScene = function () {
 8
 9    // This creates a basic Babylon Scene
          object (non-mesh)
10    var scene = new BABYLON.Scene(engine);
11
```

```
12    // This creates and positions a free
          camera (non-mesh)
13    var camera = new BABYLON.FreeCamera("
          camera1",
14      new BABYLON.Vector3(0, 5, -10),
            scene);
15    // This targets the camera to scene
          origin
16    camera.setTarget(BABYLON.Vector3.Zero
          ());
17    // This attaches the camera to the
          canvas
18    camera.attachControl(canvas, true);
19
20    // This creates a light, aiming 0,1,0
          - to the sky (non-mesh)
21    var light = new BABYLON.
          HemisphericLight("light1",
22      new BABYLON.Vector3(0, 1, 0), scene)
            ;
23    // Default intensity is 1. Let's dim
          the light a small amount
24    light.intensity = 0.7;
25
26    // Materials
27    var mat0 = new BABYLON.
          StandardMaterial('mat0', scene);
```

```
28    mat0.diffuseColor = new BABYLON.Color3
          (1, 1, 0);
29
30    var mat1 = new BABYLON.
          StandardMaterial('mat1', scene);
31    mat1.diffuseColor = new BABYLON.Color3
          (1, 0, 0);
32
33    var mat2 = new BABYLON.
          StandardMaterial('mat2', scene);
34    mat2.diffuseColor = new BABYLON.Color3
          (0, 1, 0);
35
36    // Our built-in 'ground' shape. Params
          : name, width, depth,
37    // subdivs, scene
38    var ground = BABYLON.Mesh.CreateGround
          ("ground1", 6, 6, 2, scene);
39    ground.material = mat0
40
41    var sphere = BABYLON.MeshBuilder.
          CreateSphere("sphere1",
42      {diameter: 2, segments: 16}, scene);
43    sphere.material = mat1;
44    sphere.position.y = 1;
45
46    var cube = BABYLON.MeshBuilder.
          CreateBox("cube",
```

```
47      { size: 1, height: 3 }, scene);
48    cube.position = new BABYLON.Vector3(1,
          1.5, 0);
49    cube.material = mat2;
50
51    // Merge sphere and cube into one mesh
52    var mergedMesh = BABYLON.Mesh.
          MergeMeshes([sphere, cube],
53      true, true, undefined, false, true);
54    mergedMesh.name = "mergedMesh";  //
          Assign a name to the merged mesh
55    mergedMesh.position = new BABYLON.
          Vector3(0, 0, 0);
56
57    return scene;
58  };
59
60  // Call the createScene function
61  var scene = createScene();
62
63  // Set initial position for animation
64  var animationSpeed = 0.01; // Speed of
        the movement
65
66  // Render loop with mesh animation
67  engine.runRenderLoop(function () {
68    scene.render();
69
```

```
70    // Animate merged mesh movement
          upwards
71    var mergedMesh = scene.getMeshByName("
          mergedMesh");  // Get the merged
          mesh by name
72    if (mergedMesh) {
73      mergedMesh.position.y +=
            animationSpeed;  // Move the
            merged mesh upwards
74    }
75  });
76
77  // Resize the engine if the window is
        resized
78  window.addEventListener('resize',
        function () {
79    engine.resize();
80  });
```

CHAPTER 6

# Animasi Kotak Menggelinding

**Berikut adalah contoh Minimal Working Example (MWE) menggunakan ‘scene.registerAfterRender‘ di Babylon.js untuk menampilkan sebuah **box yang menggelinding** ke kanan.**

## 6.1 HTML (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="
        width=device-width,
        initial-scale=1.0">
    <title>BabylonJS MWE</title>
    <script src="https://cdn.babylonjs.
        com/babylon.js"></script>
    <script src="app.js" defer></script>
</head>
<body>
```

```
11      <canvas id="renderCanvas" style="
            width: 100%; height: 100%;"><
            /canvas>
12  </body>
13  </html>
```

## 6.2 JavaScript (app.js)

```
1   // Get the canvas element
2   var canvas = document.getElementById('
        renderCanvas');
3
4   // Create BabylonJS engine
5   var engine = new BABYLON.Engine(canvas,
        true);
6
7   // Create the scene
8   var createScene = function () {
9       var scene = new BABYLON.Scene(engine
            );
10
11      // Create a free camera and set its
            position
12      var camera = new BABYLON.FreeCamera(
            "camera1", new BABYLON.Vector3
            (0, 5, -10), scene);
13      camera.setTarget(BABYLON.Vector3.
            Zero());  // Target the camera
```

```
        to the scene origin
14      camera.attachControl(canvas, true);
         // Allow the user to move the
        camera
15
16      // Add a hemispheric light
17      var light = new BABYLON.
        HemisphericLight("light", new
        BABYLON.Vector3(0, 1, 0), scene)
        ;
18      light.intensity = 0.7;
19
20      // Create a ground plane
21      var ground = BABYLON.MeshBuilder.
        CreateGround("ground", {width:
        10, height: 10}, scene);
22
23      // Create a box
24      var box = BABYLON.MeshBuilder.
        CreateBox("box", {size: 1},
        scene);
25      box.position.y = 0.5;  // Position
        it above the ground slightly
26
27      // Register afterRender loop to
        animate the box
28      var speed = 0.05;  // Speed of the
        box movement and rotation
```

```
29      scene.registerAfterRender(function
            () {
30          // Move the box to the right
31          box.position.x += speed;
32
33          // Rotate the box around the Z-
                axis to simulate rolling
34          box.rotation.z -= speed;
35      });
36
37      return scene;
38  };
39
40  // Call the createScene function
41  var scene = createScene();
42
43  // Start the render loop
44  engine.runRenderLoop(function () {
45      scene.render();
46  });
47
48  // Resize the engine if the window is
        resized
49  window.addEventListener('resize',
        function () {
50      engine.resize();
51  });
```

## 6.3 Penjelasan

**Box:** Dibuat menggunakan 'BABYLON.MeshBuilder.CreateBox', ditempatkan sedikit di atas tanah pada sumbu Y ('box.position.y = 0.5') agar terlihat seperti berada di atas permukaan tanah.

**Animasi dengan 'scene.registerAfterRender':** Kita menggunakan 'scene.registerAfterRender' untuk mendaftarkan sebuah fungsi yang dipanggil setelah setiap frame di-render. Di dalam fungsi ini, kotak akan bergerak ke kanan dengan menambah nilai 'x' pada posisi kotak ('box.position.x += speed'). Selain itu, kotak juga diputar pada sumbu Z ('box.rotation.z -= speed') untuk memberikan efek menggelinding.

**Ground:** Sebuah lantai atau tanah diciptakan untuk menambah visualisasi objek yang menggelinding di atas permukaan.

**Penggunaan 'runRenderLoop':** Fungsi 'engine.runRenderLoop()' digunakan untuk terus-menerus merender adegan dan memutar animasi.

**Latihan 1: Cobalah**

Gelinding Dengan setup ini, kotak akan bergerak dan "menggelinding" ke kanan saat animasi berjalan. Coba ubah nilai 'speed' dan lihatlah kecepatan atau arah gelindingnya.

CHAPTER 7

# Rotasi

Untuk rotasi objek di Babylon.js, baik menggunakan pivot maupun parent bisa efektif, tergantung pada kebutuhan spesifik animasi atau transformasi yang diinginkan:

**Pivot** digunakan ketika Anda ingin mengubah titik pusat rotasi objek tanpa mengubah hierarki objek. Jadi, rotasi akan terjadi relatif terhadap titik yang telah Anda set sebagai pivot.

**Parent** lebih tepat digunakan ketika Anda ingin melakukan transformasi terkait beberapa objek sekaligus. Objek-objek yang menjadi child akan mengikuti transformasi dari parent.

**Pivot dan Merge** berguna bila beberapa objek menggunakan metode mergeMeshes di Babylon.js dan tetap menggunakan pivot untuk rotasi. Dengan cara ini, objek-objek tersebut akan menjadi satu mesh, dan Anda bisa mengatur titik pivot untuk keseluruhan mesh tersebut.

## 7.1 Menggunakan Pivot

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>BABYLONJS Pivot Example</title>
  <style>
    canvas { width: 100%; height: 100% }
  </style>
  <script src="https://cdn.babylonjs.com
      /babylon.js"></script>
</head>
<body>
<canvas id="renderCanvas"></canvas>

<script>
  // Create the scene
  var canvas = document.getElementById("
      renderCanvas");
  var engine = new BABYLON.Engine(canvas
      , true);
  var createScene = function () {
    var scene = new BABYLON.Scene(engine
        );
    var camera = new BABYLON.
        ArcRotateCamera("camera",
      Math.PI / 2, Math.PI / 4, 5,
          BABYLON.Vector3.Zero(), scene)
          ;
```

```
22      camera.attachControl(canvas, true);
23      var light = new BABYLON.
            HemisphericLight("light",
24        new BABYLON.Vector3(1, 1, 0),
              scene);
25
26      // Create a box
27      var box = BABYLON.MeshBuilder.
            CreateBox("box",
28        {size: 1}, scene);
29      box.position.y = 1;
30
31      // Set pivot to the bottom of the
            box
32      box.setPivotPoint(new BABYLON.
            Vector3(0, -0.5, 0));
33
34      // Animation loop
35      scene.registerBeforeRender(function
            () {
36        box.rotation.y += 0.01;
37        // Rotate around the Y-axis with
              pivot at the bottom
38      });
39
40      return scene;
41    };
42    var scene = createScene();
```

```
43    engine.runRenderLoop(function () {
44        scene.render();
45    });
46 </script>
47 </body>
48 </html>
```

**Rotasi objek terjadi relatif terhadap titik pivot yang diatur (lihat baris nomor 32), tanpa mempengaruhi objek lain. Video terkait bisa dilihat di** https://youtube.com/shorts/9A40lGKDixg?si=jkUn2muB8rycLrL3

## 7.2 Menggunakan Parent

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <title>BABYLONJS Parent Example</title
         >
6    <style>
7      canvas { width: 100%; height: 100% }
8    </style>
9    <script src="https://cdn.babylonjs.com
         /babylon.js"></script>
10 </head>
11 <body>
12 <canvas id="renderCanvas"></canvas>
13
```

```
14 <script>
15   // Create the scene
16   var canvas = document.getElementById("
         renderCanvas");
17   var engine = new BABYLON.Engine(canvas
         , true);
18   var createScene = function () {
19     var scene = new BABYLON.Scene(engine
           );
20     var camera = new BABYLON.
           ArcRotateCamera("camera",
21       Math.PI / 2, Math.PI / 4, 5,
             BABYLON.Vector3.Zero(), scene)
             ;
22     camera.attachControl(canvas, true);
23     var light = new BABYLON.
           HemisphericLight("light",
24       new BABYLON.Vector3(1, 1, 0),
             scene);
25
26     // Create a parent object (empty
           mesh)
27     var parent = new BABYLON.
           TransformNode("parent", scene);
28
29     // Create a box as child
30     var box = BABYLON.MeshBuilder.
           CreateBox("box",
```

```
31        {size: 1}, scene);
32      box.position.y = 1;
33      box.parent = parent;
34
35      // Rotate the parent (which affects
            the child)
36      scene.registerBeforeRender(function
            () {
37        parent.rotation.y += 0.01;
38        // Rotate the parent, the box
              follows
39      });
40
41      return scene;
42    };
43    var scene = createScene();
44    engine.runRenderLoop(function () {
45      scene.render();
46    });
47  </script>
48  </body>
49  </html>
```

**Objek mengikuti rotasi dan transformasi dari parent-nya. Ini berguna jika Anda ingin beberapa objek bergerak secara bersama-sama. Baris 27-39 menunjukkan bahwa rotasi dilakukan pada parent (yang berupa empty mesh) sehingga child nya berputar.**

## 7.3 Menggunakan Pivot dan Merge

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>BABYLONJS Merge Meshes with Pivot</title>
  <style>
    canvas { width: 100%; height: 100% }
  </style>
  <script src="https://cdn.babylonjs.com/babylon.js"></script>
</head>
<body>
<canvas id="renderCanvas"></canvas>

<script>
  // Create the scene
  var canvas = document.getElementById("renderCanvas");
  var engine = new BABYLON.Engine(canvas, true);
  var createScene = function () {
    var scene = new BABYLON.Scene(engine);
    var camera = new BABYLON.ArcRotateCamera("camera",
```

```
21        Math.PI / 2, Math.PI / 4, 5,
              BABYLON.Vector3.Zero(), scene)
              ;
22      camera.attachControl(canvas, true);
23      var light = new BABYLON.
            HemisphericLight("light",
24        new BABYLON.Vector3(1, 1, 0),
              scene);
25
26      // Create two boxes
27      var box1 = BABYLON.MeshBuilder.
            CreateBox("box1",
28        {size: 1}, scene);
29      var box2 = BABYLON.MeshBuilder.
            CreateBox("box2",
30        {size: 1}, scene);
31
32      // Adjust positions before merging
33      box1.position = new BABYLON.Vector3(
            -1.5, 1, 0);
34      box2.position = new BABYLON.Vector3
            (1.5, 1, 0);
35
36      // Merge the boxes into a single
            mesh
37      var mergedMesh = BABYLON.Mesh.
            MergeMeshes([box1, box2],
38        true, true, undefined, false, true
```

```
            );
39
40      // Set pivot to the bottom of the
            merged mesh
41      mergedMesh.setPivotPoint(new BABYLON
            .Vector3(0, -0.5, 0));
42
43      // Animation loop
44      scene.registerBeforeRender(function
            () {
45        mergedMesh.rotation.y += 0.01;
46        // Rotate around the Y-axis using
              the pivot
47      });
48
49      return scene;
50    };
51    var scene = createScene();
52    engine.runRenderLoop(function () {
53      scene.render();
54    });
55  </script>
56  </body>
57  </html>
```

**Penjelasan:**

**Penggabungan Mesh:** Dalam contoh ini, dua kotak (box1 dan box2) digabungkan menjadi satu mesh

menggunakan fungsi BABYLON.Mesh.MergeMeshes. Ini menggabungkan dua objek menjadi satu kesatuan.

**Pivot:** Setelah mesh digabungkan, titik pivot ditetapkan di bagian bawah mesh gabungan, yaitu pada (0, -0.5, 0). Ini membuat rotasi terjadi seolah-olah di sekitar dasar mesh gabungan.

**Rotasi:** Setelah mesh digabungkan, kita memutar mesh gabungan menggunakan mergedMesh.rotation.y di sekitar sumbu-Y.

Dengan cara ini, meskipun objek-objek digabungkan, Anda tetap dapat mengontrol rotasi menggunakan pivot, dan animasi berlaku pada seluruh mesh yang sudah digabung.

## 7.4 Lebih Kompleks

### 7.4.1 File html

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="
           width=device-width,
           initial-scale=1.0">
6      <title>BabylonJS MWE</title>
```

```
    <script src="https://cdn.babylonjs.com/babylon.js"></script>
    <script src="app.js" defer></script>
</head>
<body>
    <canvas id="renderCanvas" style="width: 100%; height: 100%;"></canvas>
</body>
</html>
```

### 7.4.2 File JavaScript

```
// Get the canvas element
var canvas = document.getElementById('renderCanvas');

// Create BabylonJS engine
var engine = new BABYLON.Engine(canvas, true);

var createScene = function () {
  var scene = new BABYLON.Scene(engine);
  scene.clearColor = new BABYLON.Color3(.5, .5, .5);

  // camera
  var camera = new BABYLON.
```

```
        ArcRotateCamera("camera1", 0, 0,
        0,
13      new BABYLON.Vector3(5, 3, 0), scene)
            ;
14    camera.setPosition(new BABYLON.Vector3
        (14, 8, -12));
15    camera.attachControl(canvas, true);
16    // lights
17    var light = new BABYLON.
        HemisphericLight("light1",
18      new BABYLON.Vector3(1, 0.5, 0),
          scene);
19    light.intensity = 0.8;
20
21    // material
22    var mat = new BABYLON.StandardMaterial
        ("mat1", scene);
23    mat.alpha = 1.0;
24    mat.diffuseColor = new BABYLON.Color3
        (0.5, 0.5, 1.0);
25    mat.backFaceCulling = false;
26
27    /************Start Pilot***********/
28    var body = BABYLON.MeshBuilder.
        CreateCylinder("body",
29      { height: 0.75, diameterTop: 0.2,
          diameterBottom: 0.5,
30      tessellation: 6, subdivisions: 1 },
```

```
        scene);
var arm = BABYLON.MeshBuilder.
    CreateBox("arm", { height: 0.75,
  width: 0.3, depth: 0.1875 }, scene);
arm.position.x = 0.125;
var pilot = BABYLON.Mesh.MergeMeshes([
    body, arm], true);
/*************End Pilot***********/

/**Set Center of Rotation (CoR)
    Position, Axis and Pilot Start
    Position**/
var CoR_At = new BABYLON.Vector3(0.5,
    1.5, 1);
var axis = new BABYLON.Vector3(2, 6,
    4);
var pilotStart = new BABYLON.Vector3
    (2, 4, 4);
/*******************************/

/**Draw Axis and Sphere to show Pivot
    Position**/
var axisLine = BABYLON.MeshBuilder.
    CreateLines("axisLine",
  { points: [CoR_At.add(axis.scale
      (-50)),
  CoR_At.add(axis.scale(50))] }, scene
      );
```

```
47
48    var sphere = BABYLON.MeshBuilder.
          CreateSphere("sphere",
49      { diameter: 0.25 }, scene);
50    /*****************************/
51
52    /**********Pilot Pivot*********/
53    var pivot = new BABYLON.TransformNode(
          "root");
54    pivot.position = CoR_At;
55    pilot.parent = pivot;
56    pilot.position = pilotStart;
57    /*****************************/
58
59    /*Sphere at pivot position for viewing
          */
60    sphere.parent = pivot;
61    sphere.position = pivot.position;
62    /*****************************/
63
64
65    /**Animation of Rotation**/
66
67    /*------Animation--------*/
68    var a = 0; // for oscillation
69      var angle=0.025;
70      var axisNormal = axis.normalize();
71      scene.registerAfterRender(function()
```

```
            {
72          a +=0.005;
73          var sign = Math.cos(a)/Math.abs(
                Math.cos(a));
74          pivot.rotate(axis, angle,
                BABYLON.Space.LOCAL);
75          pivot.position = pivot.position.
                add(
76            axisNormal.scale(0.01 * sign))
                  ; //move pilot along axis
77          //sphere.position = pilot.
                position; //set pivot marker
                 position
78    });
79    /*****************************/
80
81    // show axis
82    var showAxis = function (size) {
83      var makeTextPlane = function (text,
            color, size) {
84        var dynamicTexture = new BABYLON.
              DynamicTexture(
85          "DynamicTexture", 50, scene,
                true);
86        dynamicTexture.hasAlpha = true;
87        dynamicTexture.drawText(text, 5,
              40, "bold 36px Arial",
88          color, "transparent", true);
```

```
89        var plane = new BABYLON.Mesh.
              CreatePlane("TextPlane",
90          size, scene, true);
91        plane.material = new BABYLON.
              StandardMaterial(
92          "TextPlaneMaterial", scene);
93        plane.material.backFaceCulling =
              false;
94        plane.material.specularColor = new
               BABYLON.Color3(0, 0, 0);
95        plane.material.diffuseTexture =
              dynamicTexture;
96        return plane;
97      };
98
99      var axisX = BABYLON.Mesh.CreateLines
            ("axisX", [
100       new BABYLON.Vector3.Zero(), new
              BABYLON.Vector3(size, 0, 0),
              new BABYLON.Vector3(size *
              0.95, 0.05 * size, 0),
101       new BABYLON.Vector3(size, 0, 0),
102       new BABYLON.Vector3(size * 0.95,
              -0.05 * size, 0)
103     ], scene);
104     axisX.color = new BABYLON.Color3(1,
            0, 0);
105     var xChar = makeTextPlane("X", "red"
```

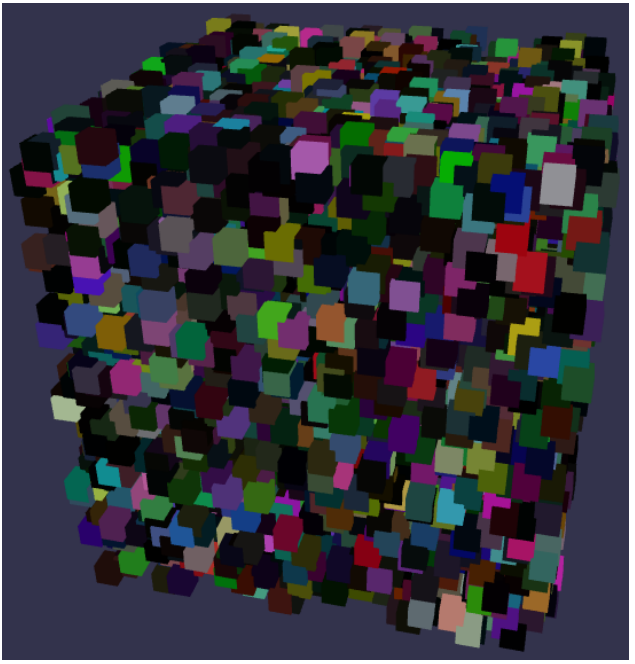
```
        , size / 10);
106     xChar.position = new BABYLON.Vector3
        (0.9 * size,
107       -0.05 * size, 0);
108     var axisY = BABYLON.Mesh.CreateLines
        ("axisY", [
109       new BABYLON.Vector3.Zero(), new
          BABYLON.Vector3(0, size, 0),
110       new BABYLON.Vector3(-0.05 * size,
          size * 0.95, 0),
111       new BABYLON.Vector3(0, size, 0),
112       new BABYLON.Vector3(0.05 * size,
          size * 0.95, 0)
113     ], scene);
114     axisY.color = new BABYLON.Color3(0,
        1, 0);
115     var yChar = makeTextPlane("Y", "
        green", size / 10);
116     yChar.position = new BABYLON.Vector3
        (0, 0.9 * size,
117       -0.05 * size);
118     var axisZ = BABYLON.Mesh.CreateLines
        ("axisZ", [
119       new BABYLON.Vector3.Zero(),
120       new BABYLON.Vector3(0, 0, size),
121       new BABYLON.Vector3(0, -0.05 *
          size, size * 0.95),
122       new BABYLON.Vector3(0, 0, size),
```

```
123         new BABYLON.Vector3(0, 0.05 * size
                , size * 0.95)
124       ], scene);
125       axisZ.color = new BABYLON.Color3(0,
              0, 1);
126       var zChar = makeTextPlane("Z", "blue
              ", size / 10);
127       zChar.position = new BABYLON.Vector3
              (0, 0.05 * size, 0.9 * size);
128     };
129
130     showAxis(8);
131     return scene;
132   };
133
134   //Call the createScene function
135   var scene = createScene();
136
137   //Render loop
138   engine.runRenderLoop(function () {
139     scene.render();
140   });
141
142   //Resize the engine if the window is
          resized
143   window.addEventListener('resize',
          function () {
144     engine.resize();
```

```
145  });
```

CHAPTER 8

# Penggandaan



## 8.1 File HTML

Program 8.1: file.html

```
1 <!DOCTYPE html>
```

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="
      width=device-width,
      initial-scale=1.0">
  <title>Babylon.js MWE - Kalimat
      Bersyarat</title>
  <script src="https://cdn.babylonjs.com
      /babylon.js"></script>
  <canvas id="renderCanvas" style="width
      : 100%; height: 100%;"></canvas>
  <script type="text/javascript" src="2.
      js"></script>
</head>
</html>
```

## 8.2 Javascript yang dipanggil HTML

**Program 8.2: 2.js**

```
// Get the canvas element
const canvas = document.getElementById("
    renderCanvas");

// Create the Babylon.js engine
const engine = new BABYLON.Engine(canvas
    , true);
```

```
6
7
8  var createScene = function() {
9    var scene = new BABYLON.Scene(engine);
10   var camera = new BABYLON.
         ArcRotateCamera("Camera", -Math.PI
          / 2, Math.PI / 2, 12, BABYLON.
         Vector3.Zero(), scene);
11   camera.attachControl(canvas, true);
12
13     var box = BABYLON.BoxBuilder.
           CreateBox("root", {size: 2});
14     box.alwaysSelectAsActiveMesh = true;
15
16     let instanceCount = 10000;
17
18     box.material = new BABYLON.
           StandardMaterial("material");
19     box.material.disableLighting = true;
20     box.material.emissiveColor = BABYLON
           .Color3.White();
21
22     box.registerInstancedBuffer("color",
            4);
23     box.instancedBuffers.color = new
           BABYLON.Color4(Math.random(),
           Math.random(), Math.random(), 1)
           ;
```

```
24
25      let baseColors = [];
26      let alphas = [];
27
28      for (var index = 0; index <
            instanceCount - 1; index++) {
29          let instance = box.
                createInstance("box" + index
                );
30          instance.position.x = 20 - Math.
                random() * 40;
31          instance.position.y = 20 - Math.
                random() * 40;
32          instance.position.z = 20 - Math.
                random() * 40;
33          instance.
                alwaysSelectAsActiveMesh =
                true;
34
35          alphas.push(Math.random());
36          baseColors.push(new BABYLON.
                Color4(Math.random(), Math.
                random(), Math.random(), 1))
                ;
37          instance.instancedBuffers.color
                = baseColors[baseColors.
                length - 1].clone();
38      }
```

```
39
40
41      scene.registerBeforeRender(() => {
42          for (var instanceIndex = 0;
                instanceIndex < box.
                instances.length;
                instanceIndex++) {
43              let alpha = alphas[
                    instanceIndex];
44              let cos = Math.abs(Math.cos(
                    alpha));
45              alpha += 0.01;
46
47              alphas[instanceIndex] =
                    alpha;
48
49              var instance = box.instances
                    [instanceIndex];
50
51              baseColors[instanceIndex].
                    scaleToRef(cos, instance
                    .instancedBuffers.color)
                    ;
52          }
53      });
54
55      scene.freezeActiveMeshes();
56
```

```
57    return scene;
58  };
59
60
61  // Call the createScene function to set
        up the scene
62  const scene = createScene();
63
64  // Render loop
65  engine.runRenderLoop(() => {
66    scene.render();
67  });
68
69  // Resize event handler
70  window.addEventListener("resize", () =>
        {
71    engine.resize();
72  });
```

## 8.3 Inti dari program

**Program 8.3: ref.js**

```
1
2  var createScene = function() {
3    var scene = new BABYLON.Scene(engine);
4    var camera = new BABYLON.
         ArcRotateCamera("Camera", -Math.PI
```

```
        / 2, Math.PI / 2, 12, BABYLON.
      Vector3.Zero(), scene);
camera.attachControl(canvas, true);

  var box = BABYLON.BoxBuilder.
      CreateBox("root", {size: 2});
  box.alwaysSelectAsActiveMesh = true;

  let instanceCount = 10000;

  box.material = new BABYLON.
      StandardMaterial("material");
  box.material.disableLighting = true;
  box.material.emissiveColor = BABYLON
      .Color3.White();

  box.registerInstancedBuffer("color",
       4);
  box.instancedBuffers.color = new
      BABYLON.Color4(Math.random(),
      Math.random(), Math.random(), 1)
      ;

  let baseColors = [];
  let alphas = [];

  for (var index = 0; index <
      instanceCount - 1; index++) {
```

```
23          let instance = box.
                createInstance("box" + index
                );
24          instance.position.x = 20 - Math.
                random() * 40;
25          instance.position.y = 20 - Math.
                random() * 40;
26          instance.position.z = 20 - Math.
                random() * 40;
27          instance.
                alwaysSelectAsActiveMesh =
                true;
28
29          alphas.push(Math.random());
30          baseColors.push(new BABYLON.
                Color4(Math.random(), Math.
                random(), Math.random(), 1))
                ;
31          instance.instancedBuffers.color
                = baseColors[baseColors.
                length - 1].clone();
32      }
33
34
35      scene.registerBeforeRender(() => {
36          for (var instanceIndex = 0;
                instanceIndex < box.
                instances.length;
```

```
                instanceIndex++) {
37              let alpha = alphas[
                    instanceIndex];
38              let cos = Math.abs(Math.cos(
                    alpha));
39              alpha += 0.01;
40
41              alphas[instanceIndex] =
                    alpha;
42
43              var instance = box.instances
                    [instanceIndex];
44
45              baseColors[instanceIndex].
                    scaleToRef(cos, instance
                    .instancedBuffers.color)
                    ;
46          }
47      });
48
49      scene.freezeActiveMeshes();
50
51    return scene;
52  };
```