# SKRIPSI

## APLIKASI PRATINJAU 3 DIMENSI BERBASIS WEB


Nancy Valentina

NPM: 2014730049

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
«tahun»

# UNDERGRADUATE THESIS

«JUDUL BAHASA INGGRIS»


Nancy Valentina

NPM: 2014730049

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY

«tahun»

# LEMBAR PENGESAHAN

**APLIKASI PRATINJAU 3 DIMENSI BERBASIS WEB**

**Nancy Valentina**

**NPM: 2014730049**

**Bandung, «tanggal» «bulan» «tahun»**

**Menyetujui,**

| Pembimbing Utama | Pembimbing Pendamping |
|---|---|
| **Pascal Alfadian, M.Comp.** | |

| Ketua Tim Penguji | Anggota Tim Penguji |
|---|---|
| «penguji 1» | «penguji 2» |

**Mengetahui,**

**Ketua Program Studi**

**Mariskha Tri Adithia, P.D.Eng**

# PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

**APLIKASI PRATINJAU 3 DIMENSI BERBASIS WEB**

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal «tanggal» «bulan» «tahun»

Meterai
Rp. 6000

Nancy Valentina
NPM: 2014730049

## ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.



**Kata-kata kunci:** «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»

## ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»
Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.



**Keywords:** «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»

*«kepada siapa anda mempersembahkan skripsi ini. . . ?»*

# KATA PENGANTAR

«Tuliskan kata pengantar dari anda di sini . . . »
Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Bandung, «bulan» «tahun»

Penulis

# DAFTAR ISI


**Kata Pengantar** xv

**Daftar Isi** xvii

**Daftar Gambar** xix

**Daftar Tabel** xxi

**1 Pendahuluan** 1
- 1.1 Latar Belakang . . . 1
- 1.2 Rumusan Masalah . . . 1
- 1.3 Tujuan . . . 3
- 1.4 Batasan Masalah . . . 3
- 1.5 Metodologi . . . 3
- 1.6 Sistematika Pembahasan . . . 3

**2 Landasan Teori** 5
- 2.1 WebGL . . . 5
- 2.2 Pustaka Three.js . . . 9

**A Kode Program** 35

**B Hasil Eksperimen** 37

# DAFTAR GAMBAR

1.1 ruangan perkuliahan di Fakultas Teknologi Informasi dan Sains (1) . . . . . 2
1.2 ruangan perkuliahan di Fakultas Teknologi Informasi dan Sains (2) . . . . . 2

B.1 Hasil 1 . . . . . 37
B.2 Hasil 2 . . . . . 37
B.3 Hasil 3 . . . . . 37
B.4 Hasil 4 . . . . . 37

# DAFTAR TABEL

# BAB 1

# PENDAHULUAN

## 1.1 Latar Belakang

Aplikasi pratinjau 3 dimensi merupakan sebuah perangkat lunak yang membantu penggunanya untuk meninjau kembali desain dari produk yang ingin dihasilkan secara 3 dimensi, sebelum pengguna tersebut melakukan implementasi pembuatan produk. Kelebihan dari aplikasi ini adalah pengguna dapat melakukan peninjauan dari berbagai sudut pandang untuk memaksimalkan hasil dari implementasi pembuatan produk. Aplikasi pratinjau tiga dimensi juga memungkinkan pengguna untuk merubah desain dari produk, hal ini bertujuan agar dapat membantu pengguna memutuskan desain produk yang paling sesuai. Pada dasarnya aplikasi pratinjau tiga dimensi bertujuan untuk membantu pengguna agar terhindar dari hasil pembuatan produk yang tidak sesuai dengan ekspektasi pengguna.

Penggunaan teknologi *web* pada aplikasi 3 dimensi dapat memudahkan pengguna untuk melakukan akses aplikasi tanpa harus melakukan instalasi aplikasi namun hanya menggunakan *browser*. Kemudian aplikasi berbasis web juga ramah untuk berbagai lingkungan sistem operasi seperti Windows, Linux, dan Mac OS sehingga tidak membatasi cakupan penggunanya.

Pada skripsi ini, akan dibuat aplikasi pratinjau 3 dimensi berbasis web yang dapat memungkinkan pengguna untuk melakukan kustomisasi ruang belajar mengajar pada lingkungan perkuliahan. Melalui perangkat lunak ini, pengguna diharapkan dapat memiliki gambaran 3 dimensi mengenai ruangan belajar mengajar dengan komposisi warna dinding dan tekstur lantai yang tepat. Perangkat lunak akan dibuat dengan memanfaatkan WebGL dan pustaka Three.js. WebGL merupakan sebuah lintas platform, standar web bebas royalti untuk *Application Programming Interface* (API) grafis 3 dimensi level rendah yang berdasar dari OpenGL ES, terbuka untuk ECMAScript melalui elemen *canvas* HTML5. Sementara itu pustaka Three.js bertujuan membuat pustaka 3 dimensi yang mudah dan ringan untuk digunakan. Kemudian sebagai studi kasus, ruangan belajar mengajar yang akan digunakan untuk melakukan simulasi aplikasi pratinjau tiga dimensi berbasis *web* adalah salah satu ruangan perkuliahan di Fakultas Teknologi Informasi dan Sains. Ruangan tersebut dilengkapi dengan peralatan multimedia yang dapat menunjang pengajaran berbasis Teknologi Informasi seperti komputer, proyektor, serta koneksi internet yang dapat menunjang perkuliahan berbasis E-learning. Selain itu untuk menjamin kenyamanan selama perkuliahan, semua ruang kuliah dilengkapi pendingin udara.

## 1.2 Rumusan Masalah

Berikut ini masalah-masalah yang dibahas dalam skripsi ini:

- Bagaimana ruangan kelas dan objek pendukung lainnya dapat direpsentasikan dalam WebGL?
- Bagaimana membuat tampilan responsif pada aplikasi agar terlihat bagus saat dicetak?

Gambar 1.1: ruangan perkuliahan di Fakultas Teknologi Informasi dan Sains (1)

Gambar 1.2: ruangan perkuliahan di Fakultas Teknologi Informasi dan Sains (2)

## 1.3 Tujuan

Berikut ini tujuan-tujuan yang ingin dicapai dalam penilitian ini:

- Membangun aplikasi yang dapat merepresentasikan ruangan dalam WebGL.
- Membangun tampilan aplikasi yang responsif sehingga terlihat bagus saat dicetak.

## 1.4 Batasan Masalah

Terdapat beberapa batasan masalah dalam penilitian ini, yaitu:

1. Pengguna hanya dapat melakukan kustomisasi pada tekstur lantai, warna cat dinding bagian atas, dan warna cat dinding bagian bawah dari ruangan kelas.
2. Pengguna hanya dapat mengganti tekstur lantai, warna cat dinding bagian atas, dan warna cat dinding bagian bawah dengan 8 variasi.

## 1.5 Metodologi

Metodologi yang digunakan untuk menyusun penelitian ini adalah sebagai berikut:

1. Mempelajari standar WebGL sebagai *Application Programming Interface* untuk menampilkan grafis 3 dimensi pada *web browser*.
2. Mempelajari penggunaan Three.js sebagai *library* dari WebGL.
3. Memodelkan ruangan belajar mengajar secara 3 dimensi.
4. Melakukan analisis terhadap situs web yang akan dibangun.
5. Merancang tampilan situs web yang akan dibangun.
6. Mengimplementasikan situs web.
7. Melakukan pengujian terhadap situs web yang telah dibangun.
8. Menulis dokumen skripsi.

## 1.6 Sistematika Pembahasan

Pembahasan dalam buku skripsi ini dilakukan secara sistematis sebagai berikut:

- Bab 1 Pendahuluan Berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi penilitian, dan sistematika pembahasan.
- Bab 2 Dasar Teori Berisi teori-teori dasar mengenai WebGL dan Three.js *library*.
- Bab 3 Analisis Berisi analisis masalah dan solusi, studi kasus, perancangan perangkat lunak, diagram aktivitas, *use case* diagram, dan diagram paket.
- Bab 4 Perancangan Berisi perancangan antarmuka dan diagram kelas.
- Bab 5 Implementasi Berisi implementasi antarmuka perangkat lunak, implementasi menggunakan WebGL dan *library* Three.js, pengujian perangkat lunak yang telah dibangun, dan kesimpulan berdasarkan pengujian.
- Bab 6 Kesimpulan dan Saran Berisi kesimpulan berdasarkan pengujian yang telah dilakukan dan saran untuk penelitian berikutnya.

# BAB 2

# LANDASAN TEORI

Bab ini berisi penjelasan mengenai teori-teori yang menjadi dasar penelitian ini, seperti WebGL dan Three.js *library*.

## 2.1 WebGL

WebGL adalah sebuah Application Programming Interface (API) yang membangun objek 3 dimensi dengan mode langsung yang dirancang untuk *web*. WebGL diturunkan dari OpenGL ES 2.0, menyediakan fungsi pembangunan sejenis tetapi di dalam konteks HTML. WebGL dirancang sebagai konteks pembangunan objek pada elemen *canvas* HTML. *Canvas* pada HTML menyediakan suatu destinasi untuk pembangunan objek secara programatik pada halaman *web* dan memungkinkan menampilkan objek yang sedang dibangun menggunakan API pembangun objek yang berbeda [**?**]. Berikut ini merupakan *interfaces* dan fungsionalitas yang ada pada WebGL:

1. *Types*

   Berikut ini merupakan tipe-tipe yang digunakan pada semua *interface* di bagian penjelasan selanjutnya. Kemudian dijelaskan juga alias untuk semua tipe yang ada pada WebGL.

```
typedef unsigned long  GLenum;
typedef boolean        GLboolean;
typedef unsigned long  GLbitfield;
typedef byte           GLbyte;
typedef short          GLshort;
typedef long           GLint;
typedef long           GLsizei;
typedef long long      GLintptr;
typedef long long      GLsizeiptr;
typedef octet          GLubyte;
typedef unsigned short GLushort;
typedef unsigned long  GLuint;
typedef unrestricted float GLfloat;
typedef unrestricted float GLclampf;
```

   Listing 2.1: type yang serupa hehe

2. *WebGLContextAttributes*

   *WebGLContextAttributes* merupakan kamus yang berisi atribut-atribut latar untuk menggambar yang diberikan melalui parameter kedua pada *getContext*. Berikut ini merupakan daftar nilai awal dari atribut pada *WebGLContextAttributes*, nilai awal ini akan digunakan apabila tidak ada parameter kedua yang diberikan kepada *getContext* atau jika objek pengguna yang tidak memiliki atribut pada namanya diberikan kepada getContext.

```
dictionary WebGLContextAttributes {
    GLboolean alpha = true;
    GLboolean depth = true;
    GLboolean stencil = false;
    GLboolean antialias = true;
    GLboolean premultipliedAlpha = true;
    GLboolean preserveDrawingBuffer = false;
    WebGLPowerPreference powerPreference = "default";
   GLboolean failIfMajorPerformanceCaveat = false;
};
```

Berikut ini merupakan penjelasan setiap atribut pada *WebGLContextAttributes*

- *alpha*

  Jika nilainya *true*, penyangga gambar telah memiliki *alpha channel* yang bertujuan untuk menampilkan operasi *alpha* destinasi OpenGL . Jika nilainya *false*, tidak ada penyangga *alpha* yang tersedia.

- *depth*

  Jika nilainya *true*, penyangga gambar memiliki sebuah penyangga kedalaman yang setidaknya berisi 16 *bits*. Jika nilainya *false*, tidak ada penyangga kedalaman yang tersedia.

- *stencil*

  Jika nilainya *true*, penyangga gambar memiliki penyangga stensil yang setidaknya berisi 8 *bits*. Jika nilainya *false*, tidak ada penyangga stensil yang tersedia.

- *antialias*

  Jika nilainya *true* dan implementasinya mendukung *antialias* maka penyangga gambar akan menampilkan *antialias* menggunakan teknik yang dipilih dan kualitas. Jika nilainya *false* atau implementasi tidak mendukung *antialias* maka tidak ada *antialias* yang ditampilkan.

- *premultipliedAlpha*

  Jika nilainya *true*, penyusun halaman akan mengasumsikan penyangga gambar memiliki warna dengan *premultiplied alpha*. Jika nilainya *false*, penyusun halaman akan mengasumsikan bahwa warna pada penyangga gambar bukan *premultiplied*.

- *preserveDrawingBuffer*

  Jika nilainya *false* saat penyangga gambar mempresentasikan bagian dari penyangga gambar yang terdeskripsikan, konten-konten pada penyangga gambar akan dihapus ke nilai awalnya. Begitupun jug adengan elemen dari penyangga gambar seperti warna, kedalaman, dan stensil yang juga akan dihapus. Jika nilainya *true*, penyangga tidak akan dihapus dan akan mempresentasikan nilainya sampai nantinya dihapus atau ditulis kembali oleh penulisnya.

- *powerPreference*

  Menyediakan petunjuk untuk agen pengguna yang mengindikasikan konfigurasi GPU yang cocok untuk konteks WebGL tersebut.

- *failIfMajorPerformanceCaveat*

  Jika nilainya *true*, pembuatan konteks akan gagal jika implementasi menentukan bahwa performansi pada konteks WebGL yang dibuat akan sangat rendah pada aplikasi yang membuat persamaan pemanggilan OpenGL.

3. *WebGLObject*

   *Interface WebGLObject* merupakan *interface* awal untuk diturunkan kepada semua objek GL.

```
interface WebGLObject {
};
```

4. *WebGLBuffer*

   *Interface WebGLBuffer* merepresentasikan sebuah OpenGL *Buffer Object*.

   ```
   interface WebGLBuffer : WebGLObject {
   };
   ```

5. *WebGLFrameBuffer*

   *Interface WebGLFrameBuffer* merepresentasikan sebuah OpenGL *Frame Buffer Object*.

   ```
   interface WebGLFramebuffer : WebGLObject {
   };
   ```

6. *WebGLProgram*

   *Interface WebGLProgram* merepresentasikan sebuah OpenGL *Program Object*.

   ```
   interface WebGLProgram : WebGLObject {
   };
   ```

7. *WebGLRenderbuffer*

   *Interface WebGLRenderbuffer* merepresentasikan sebuah OpenGL *Renderbuffer Object*.

   ```
   interface WebGLRenderbuffer : WebGLObject {
   };
   ```

8. *WebGLShader*

   *Interface WebGLShader* merepresentasikan sebuah OpenGL *Shader Object*.

   ```
   interface WebGLShader : WebGLObject {
   };
   ```

9. *WebGLTexture*

   *Interface WebGLTexture* merepresentasikan sebuah OpenGL *Texture Object*.

   ```
   interface WebGLTexture : WebGLObject {
   };
   ```

10. *WebGLUniformLocation*

    *Interface WebGLUniformLocation* merepresentasikan lokasi dari variabel *uniform* pada program *shader*.

    ```
    interface WebGLUniformLocation {
    };
    ```

11. *WebGLActiveInfo*

    *Interface WebGLActiveInfo* merepresentasikan informasi yang dikembalikan dari pemanggilan *getActiveAttrib* dan *getActiveUniform*.

    ```
    interface WebGLActiveInfo {
            readonly attribute GLint size;
            readonly attribute GLenum type;
            readonly attribute DOMString name;
    };
    ```

12. *WebGLShaderPrecisionFormat*

    *Interface WebGLShaderPrecisionFormat* merepresentasikan informasi yang dikembalikan dari pemanggilan *getShaderPrecisionFormat*.

```
interface WebGLShaderPrecisionFormat {
        readonly attribute GLint rangeMin;
        readonly attribute GLint rangeMax;
        readonly attribute GLint precision;
};
```

13. *ArrayBuffer* dan *Typed Arrays*

    *Vertex, index, texture,* dan data lainnya ditransfer ke implementasi WebGL menggunakan *ArrayBuffer, Typed Arrays,* dan *DataViews* seperti yang telah didefinisikan pada spesifikasi ECMAScript.

```
var numVertices = 100; // for example

// Hitung ukuran buffer yang dibutuhkan dalam bytes dan floats
var vertexSize = 3 * Float32Array.BYTES_PER_ELEMENT +
4 * Uint8Array.BYTES_PER_ELEMENT;
var vertexSizeInFloats = vertexSize / Float32Array.BYTES_PER_ELEMENT;

// Alokasikan buffer
var buf = new ArrayBuffer(numVertices * vertexSize);

// Map buffer ke Float32Array untuk mengakses posisi
var positionArray = new Float32Array(buf);

        // Map buffer yang sama ke Uint8Array untuk mengakses warna
var colorArray = new Uint8Array(buf);

// Inisialisasi offset dari vertices dan warna pada buffer
var positionIdx = 0;
var colorIdx = 3 * Float32Array.BYTES_PER_ELEMENT;

// Inisialisasi buffer
for (var i = 0; i < numVertices; i++) {
        positionArray[positionIdx] = ...;
        positionArray[positionIdx + 1] = ...;
        positionArray[positionIdx + 2] = ...;
        colorArray[colorIdx] = ...;
        colorArray[colorIdx + 1] = ...;
        colorArray[colorIdx + 2] = ...;
        colorArray[colorIdx + 3] = ...;
        positionIdx += vertexSizeInFloats;
        colorIdx += vertexSize;
}
```

14. *WebGL Contect WebGLRenderingContext* merepresentasikan API yang memungkinkan gaya pembangunan OpenGL ES 2.0 ke elemen *canvas*.

15. *WebGLContextEvent* WebGL menghasilkan sebuah *WebGLContextEvent* sebagai respon dari perubahan penting pada status konteks pembangunan WebGL. *Event* tersebut dikirim melalui

*DOM Event System* dan dilanjutkan ke HTMLCanvasEvent yang diasosiasikan dengan konteks pembangunan WebGL.

## 2.2 Pustaka Three.js

Pustaka Three.js ini bertujuan untuk membuat pustaka 3 dimensi yang mudah dan ringan untuk digunakan. Pustaka ini menyediakan <canvas>, <svg>, dan CSS3D, dan pembangun WebGL [**?**]. Terdapat beberapa fungsi penting yang disediakan oleh pustaka Three.js dalam pembuatan grafis 3 dimensi, di antaranya adalah [**?**]:

- *Cameras*
  - *Camera*, kelas abstrak untuk *cameras*. Kelas ini harus selalu diwarisi saat membangun suatu kamera. Konstruktor pada kelas ini digunakan untuk membuat kamera baru, namun kelas ini tidak dipergunakan secara langsung melainkan menggunakan *PerspectiveCamera* atau *OrthographicCamera*.
  - *CubeCamera*, membuat 6 kamera yang dibangun pada *WebGLRenderTargetCube*. Konstruktor pada kelas ini menerima parameter berupa jarak terdekat, jarak terjauh, dan resolusi dari kubus.

    ```
    var cubeCamera = new THREE.CubeCamera( 1, 100000, 128 );
    scene.add( cubeCamera );
    ```

    Listing 2.2: Contoh instansiasi kelas *CubeCamera*.

  - *OrthographicCamera*, kamera yang menggunakan proyeksi ortografik. Konstruktor pada kelas ini menerima parameter berupa *frustum* kamera bagian kiri, *frustum* kamera bagian kanan, *frustum* kamera bagian atas, *frustum* kamera bagian bawah, *frustum* kamera untuk jarak dekat, dan *frustum* kamera untuk jarak jauh.

    ```
    var camera = new THREE.OrthographicCamera( width / - 2, width / 2,
    height / 2, height / - 2, 1, 1000 );
    scene.add( camera );
    ```

    Listing 2.3: Contoh instansiasi kelas *OrthographicCamera*

  - *PerspectiveCamera*, kamera yang menggunakan pyoyeksi perspektif. Konstruktor pada kelas ini menerima parameter berupa *frustum* pandangan vertikal, *frustum* pandangan horizontal, dan *frustum* jarak dekat, dan *frustum* jarak jauh.

    ```
    var camera = new THREE.PerspectiveCamera( 45, width / height,
    1, 1000 );
    scene.add( camera );
    ```

    Listing 2.4: Contoh instansiasi kelas *PerspectiveCamera*

  - *StereoCamera*, dua buah *PerspektifCamera* yang digunakan untuk efek seperti *3D Anaglyph* dan *Parallax Barrier*.
- *Core*
  - *BufferAttribute*, kelas ini menyimpan data untuk atribut yang diasosiasikan menggunakan BufferGeometry. Hal ini memungkinkan pengiriman data yang lebih efisien kepada GPU. Konstruktor pada kelas ini menerima parameter berupa sebuah array dengan ukuran nilai dari array dikalikan dengan jumlah vertex, nilai dari array tersebut, dan juga sebuah boolean yang merepresentasikan penggunaan *normalized*.

- *BufferGeometry*, merupakan sebuah kelas alternatif efisien untuk *Geometry*. Karena kelas ini menyimpan semua data, termasuk posisi vertex, index permukaan, normal, warna, UV, dan atribut kustom menggunakan buffer. Kelas ini mengurangi biaya pengiriman seluruh data ke GPU. Konstruktor pada kelas ini digunakan untuk membuat *BufferGeometry* baru dan inisialisasi nilai awal untuk objek baru tersebut.

```
var geometry = new THREE.BufferGeometry();
// membuat bentuk kotak sederhana dengan melakukan duplikasi pada
// bagian atas kiri dan bawah kanan
// kumpulan vertex karena setiap vertex harus muncul di setiap segitiga
var vertices = new Float32Array( [
        -1.0, -1.0,  1.0,
         1.0, -1.0,  1.0,
         1.0,  1.0,  1.0,

         1.0,  1.0,  1.0,
        -1.0,  1.0,  1.0,
        -1.0, -1.0,  1.0
] );

// itemSize = 3 karena ada 3 values (components) per vertex
geometry.addAttribute( 'position', new THREE.BufferAttribute
( vertices, 3 ) );
var material = new THREE.MeshBasicMaterial( { color: 0xff0000 } );
var mesh = new THREE.Mesh( geometry, material );
```

Listing 2.5: Contoh instansiasi kelas *BufferGeometry* dengan membuat bentuk kotak sederhana.

- *Clock*, sebuah objek untuk menjaga alur dari waktu.
- Direct Geometry, kelas ini digunakan secara internal untuk mengkonversi *Geometry* menjadi *BufferGeometry*. Konstruktor pada kelas ini digunakan untuk membuat *DirectGeometry* baru.
- *EventDispatcher*, suatu *event* pada JavaScript untuk objek kustom. Konstruktor pada kelas ini digunakan untuk membuat objek *EventDispatcher*.

```
// menambahkan event untuk objek kustom
var Car = function () {
    this.start = function () {
        this.dispatchEvent( { type: 'start',
        message: 'vroom vroom!' } );
    };
};

//  mencampur EventDispatcher.prototype dengan prototipe objek kustom
Object.assign( Car.prototype, EventDispatcher.prototype );

// Using events with the custom object

var car = new Car();

car.addEventListener( 'start', function ( event ) {

    alert( event.message );
```

```
} );

car.start();
```

Listing 2.6: Contoh penggunaan objek *EventDispatcher* untuk objek kustom.

– *Face3*, permukaan segitiga yang digunakan pada *Geometry*. Konstruktor pada kelas ini menerima parameter berupa vertek A, vertek B, vertek C, sebuah vektor permukaan normal atau *array* dari vertek normal, sebuah warna permukaan atau *array* dari vertek warna, dan indeks dari *array* material yang akan diasosiasikan dengan permukaan.

```
var material = new THREE.MeshStandardMaterial( { color : 0x00cc00 } );

// membuat geometry segitiga
var geometry = new THREE.Geometry();
geometry.vertices.push( new THREE.Vector3( -50, -50, 0 ) );
geometry.vertices.push( new THREE.Vector3(  50, -50, 0 ) );
geometry.vertices.push( new THREE.Vector3(  50,  50, 0 ) );

//membuat permukaan baru dengan vertex 0, 1, 2
var normal = new THREE.Vector3( 0, 1, 0 ); //optional
var color = new THREE.Color( 0xffaa00 ); //optional
var materialIndex = 0; //optional
var face = new THREE.Face3( 0, 1, 2, normal, color, materialIndex );

// menambahkan permukaan ke array permukaan geometry
geometry.faces.push( face );

// permukaan normal dan vertex normal dapat dihitung
// secara otomatis apabila tidak disediakan di atas
geometry.computeFaceNormals();
geometry.computeVertexNormals();

scene.add( new THREE.Mesh( geometry, material ) );
```

Listing 2.7: Contoh penggunaan *Face3* pada suatu *Geometry*.

– *Geometry*, kelas dasar untuk *Geometry*.

```
var geometry = new THREE.Geometry();

geometry.vertices.push(
        new THREE.Vector3( -10,  10, 0 ),
        new THREE.Vector3( -10, -10, 0 ),
        new THREE.Vector3(  10, -10, 0 )
);

geometry.faces.push( new THREE.Face3( 0, 1, 2 ) );

geometry.computeBoundingSphere();
```

Listing 2.8: Contoh instansiasi kelas *Geometry*.

– *InstancedBufferAttribute*, sebuah versi instansi dari *BufferAttribute*. Konstruktor pada kelas ini menerima parameter berupa sebuah array dengan ukuran nilai dari array

dikalikan dengan jumlah vertex, nilai dari array tersebut, dan juga jumlah jala pada setiap atribut dengan nilai awal adalah 1.

- *InstancedBufferGeometry*, sebuah versi instansi dari *BufferGeometry*.
- *InstancedInterleavedBuffer*, sebuah versi instansi dari *InterleavedBuffer*. Konstruktor pada kelas ini menerima parameter berupa sebuah array dengan ukuran nilai dari array dikalikan dengan jumlah vertex, nilai dari array tersebut, dan juga jumlah jala pada setiap atribut dengan nilai awal adalah 1.
- *InterleavedBuffer*. Konstruktor pada kelas ini menerima parameter berupa sebuah *array* dan *stride*.
- *InterleavedBufferAttribute*. Konstruktor pada kelas ini menerima parameter berupa sebuah objek *InterleavedBuffer*, ukuran benda, *offset*, dan sebuah boolean yang merepresentasikan *normalized* dengan nilai awal adalah *true*.
- *Layers*, lapisan-lapisan objek yang berisi dari objek 3 dimensi dan terdiri dari 1 sampai 32 layer yang diberi nomor 0 sampai 31. Secara internal, layer disimpan sebagai sebuah *bit mask*. Kemudian sebagai inisialisasinya, semua anggota dai *Object3Ds* merupakan member dari lapisan 0. Konstruktor pada kelas ini digunakan untuk membuat objek *Layers* baru dengan anggota awal berada pada lapisan 0.
- *Object3D*, sebuah kelas dasar untuk hampir semua object pada Three.js yang juga menyediakan seperangkat properti dan metode untuk memanipulasi objek 3 dimenasi pada ruang.
- *Raycaster*, sebuah kelas yang didesain untuk membantu *raycasting*. *Raycasting* digunakan untuk mengetahui posisi kursor berada pada suatu benda diantara benda lainnya. Konstruktor pada kelas ini menerima parameter berupa vektor awal asal sinar, arah sinar, jarak terdekat, dan jarak terjauh.

```
var raycaster = new THREE.Raycaster();
var mouse = new THREE.Vector2();

function onMouseMove( event ) {
        // menghitung posisi kursor pada koordinat perangkat normal
        // (-1 to +1) untuk kedua komponen

        mouse.x = ( event.clientX / window.innerWidth ) * 2 - 1;
        mouse.y = - ( event.clientY / window.innerHeight ) * 2 + 1;
}

function render() {
        // mengubah sinar dari kamera dan posisi kursor
        raycaster.setFromCamera( mouse, camera );

        // kalkulasi objek yang berpotongan pada sinar
        var intersects = raycaster.intersectObjects( scene.children );

        for ( var i = 0; i < intersects.length; i++ ) {
                intersects[ i ].object.material.color.set( 0xff0000 );
        }
        renderer.render( scene, camera );
}

window.addEventListener( 'mousemove', onMouseMove, false );
```

```
window.requestAnimationFrame(render);
```

Listing 2.9: Contoh penggunaan kelas *Raycaster*.

– *Uniform*, merupakan variabel global GLSL. *Uniform* akan dikirim ke program *shader*.

```
uniforms: {
        time: { value: 1.0 },
        resolution: new THREE.Uniform(new THREE.Vector2())
}
```

Listing 2.10: Contoh penggunaan kelas *Uniform* yang diinisialisasi dengan nilai atau objek.

- *Geometries*

  – *BoxBufferGeometry*, merupakan port *BufferGeometry* dari *BoxGeometry*. Konstruktor pada kelas ini menerima parameter berupa lebar sisi pada sumbu X dengan nilai awal adalah 1, tinggi sisi pada sumbu Y dengan nilai awal adalah 1, kedalaman sisi pada sumbu Z dengan nilai awal adalah 1, jumlah permukaan yang berpotongan dengan lebar sisi dengan nilai awal adalah 1 dan bersifat fakultatif, jumlah permukaan yang berpotongan dengan tinggi sisi dengan nilai awal adalah 1 dan bersifat fakultatif, dan jumlah permukaan yang berpotongan dengan kedalaman sisi dengan nilai awal adalah 1 dan bersifat fakultatif.

```
var geometry = new THREE.BoxBufferGeometry( 1, 1, 1 );
var material = new THREE.MeshBasicMaterial( {color: 0x00ff00} );
var cube = new THREE.Mesh( geometry, material );
scene.add( cube );
```

Listing 2.11: Contoh penggunaan kelas *BoxBufferGeometry*.

  – *BoxGeometry*, merupakan kelas primitif geometri berbentuk segi empat. Contoh penggunaannya sama seperti kelas *BoxBufferGeometry*. Konstruktor pada kelas ini menerima parameter berupa lebar sisi pada sumbu X dengan nilai awal adalah 1, tinggi sisi pada sumbu Y dengan nilai awal adalah 1, kedalaman sisi pada sumbu Z dengan nilai awal adalah 1, jumlah permukaan yang berpotongan dengan lebar sisi dengan nilai awal adalah 1 dan bersifat fakultatif, jumlah permukaan yang berpotongan dengan tinggi sisi dengan nilai awal adalah 1 dan bersifat fakultatif, dan jumlah permukaan yang berpotongan dengan kedalaman sisi dengan nilai awal adalah 1 dan bersifat fakultatif.

  – *CircleBufferGeometry*, merupakan port *BufferGeometry* dari *CircleGeometry*. Konstruktor pada kelas ini menerima parameter berupa radius lingkaran dengan nilai awal adalah 50, jumlah banyak bagian dengan minimum adalah 3 dan nilai awal adalah 8, sudut dimulainya bagian pertama dengan nilai awal adalah 0, dan sudut pusat dengan nilai awal adalah 2 kali Pi. Contoh penggunaannya sama seperti kelas *BoxBufferGeometry*.

  – *CircleGeometry*, merupakan bentuk sederhana dari geometri *Euclidean*. Konstruktor pada kelas ini menerima parameter berupa radius lingkaran dengan nilai awal adalah 50, jumlah banyak bagian dengan minimum adalah 3 dan nilai awal adalah 8, sudut dimulainya bagian pertama dengan nilai awal adalah 0, dan sudut pusat dengan nilai awal adalah 2 kali Pi. Contoh penggunaannya sma seperti kelas *BoxBufferGeometry*.

  – *ConeBufferGeometry*, merupakan port *BufferGeometry* dari *ConeGeometry*. Konstruktor pada kelas ini menerima parameter berupa radius lingkaran untuk dasar kerucut dengan nilai awal adalah 20, tinggi kerucut dengan nilai awal adalah 100, jumlah banyak permukaan bagian dengan nilai awal adalah 8, banyak baris permukaan berdasarkan tinggi kerucut dengan nilai awal adalah 1, sebuah boolean yang menyatakan dasar kerucut

tertutup atau terbuka, sudut dimulainya bagian pertama dengan nilai awal adalah 0, dan sudut pusat dengan nilai awal adalah 2 kali Pi. Contoh penggunaannya sama seperti kelas *BoxBufferGeometry*.

- *ConeGeometry*, sebuah kelas untuk mengeneralisasi geometri kerucut. Konstruktor pada kelas ini menerima parameter berupa radius lingkaran untuk dasar kerucut dengan nilai awal adalah 20, tinggi kerucut dengan nilai awal adalah 100, jumlah banyak permukaan bagian dengan nilai awal adalah 8, banyak baris permukaan berdasarkan tinggi kerucut dengan nilai awal adalah 1, sebuah boolean yang menyatakan dasar kerucut tertutup atau terbuka, sudut dimulainya bagian pertama dengan nilai awal adalah 0, dan sudut pusat dengan nilai awal adalah 2 kali Pi. Contoh penggunaannya sama seperti kelas *BoxBufferGeometry*.
- *CylinderBufferGeometry*, merupakan port *BufferGeometry* dari *CylinderGeometry*. Konstruktor pada kelas ini menerima parameter berupa radius dari lingkaran bagian atas dengan nilai awal adalah 20, radius dari lingkaran bagian bawah dengan nilai awal adalah 20, tinggi silinder dengan nilai awal adalah 100, jumlah banyak permukaan bagian dengan nilai awal adalah 8, banyak baris permukaan berdasarkan tinggi silinder dengan nilai awal adalah 1, sebuah boolean yang menyatakan dasar silinder tertutup atau terbuka, sudut dimulainya bagian pertama dengan nilai awal adalah 0, dan sudut pusat dengan nilai awal adalah 2 kali Pi. Contoh penggunaannya sama seperti kelas *BoxBufferGeometry*.
- *CylinderGeometry*, Konstruktor pada kelas ini menerima parameter berupa radius dari lingkaran bagian atas dengan nilai awal adalah 20, radius dari lingkaran bagian bawah dengan nilai awal adalah 20, tinggi silinder dengan nilai awal adalah 100, jumlah banyak permukaan bagian dengan nilai awal adalah 8, banyak baris permukaan berdasarkan tinggi silinder dengan nilai awal adalah 1, sebuah boolean yang menyatakan dasar silinder tertutup atau terbuka, sudut dimulainya bagian pertama dengan nilai awal adalah 0, dan sudut pusat dengan nilai awal adalah 2 kali Pi. sebuah kelas untuk mengeneralisasi geometri silinder. Contoh penggunaannya sama seperti kelas *BoxBufferGeometry*.
- *DodecahedronBufferGeometry*, sebuah kelas untuk mengeneralisasi geometri pigura berduabelas segi. Konstruktor pada kelas ini menerima parameter berupa radius dari pigura berduabelas segi dengan nilai awal adalah 1 dan detail dengan nilai awal adalah 1.
- *DodecahedronGeometry*, sebuah kelas untuk mengeneralisasi geometri pigura berduabelas segi. Konstruktor pada kelas ini menerima parameter berupa radius dari pigura berduabelas segi dengan nilai awal adalah 1 dan detail dengan nilai awal adalah 1.
- *EdgesGeometry*, dapat digunakan sebagai objek pembantu untuk melihat tepi dari suatu objek geometri. Konstruktor pada kelas ini menerima parameter berupa objek geometri dan tepi sudut dengan nilai awal adalah 1.

```
var geometry = new THREE.BoxBufferGeometry( 100, 100, 100 );
var edges = new THREE.EdgesGeometry( geometry );
var line = new THREE.LineSegments( edges,
new THREE.LineBasicMaterial( { color: 0xffffff } ) );
scene.add( line );
```

Listing 2.12: Contoh penggunaan kelas *EdgesGeometry*.

- *ExtrudeGeometry*, membuat geometri diekstrusi dari sebuah alur bentuk. Konstruktor pada kelas ini menerima parameter berupa bentuk atau *array* dari bentuk dan juga pilihan yang dapat berisi beberapa parameter seperti jumlah titik pada lengkungan, jumlah titik yang digunakan untuk membagi potongan, dan lain-lain.

```
var length = 12, width = 8;
```

```
var shape = new THREE.Shape();
shape.moveTo( 0,0 );
shape.lineTo( 0, width );
shape.lineTo( length, width );
shape.lineTo( length, 0 );
shape.lineTo( 0, 0 );

var extrudeSettings = {
        steps: 2,
        amount: 16,
        bevelEnabled: true,
        bevelThickness: 1,
        bevelSize: 1,
        bevelSegments: 1
};

var geometry = new THREE.ExtrudeGeometry( shape, extrudeSettings );
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
var mesh = new THREE.Mesh( geometry, material ) ;
scene.add( mesh );
```

Listing 2.13: Contoh penggunaan kelas *ExtrudeGeometry*.

- *ExtrudeBufferGeometry*, membuat *BufferGeometry* diekstrusi dari sebuah alur bentuk. Contoh penggunaannya sama seperti kelas *ExtrudeGeometry*. Konstruktor pada kelas ini menerima parameter berupa bentuk atau *array* dari bentuk dan juga pilihan yang dapat berisi beberapa parameter seperti jumlah titik pada lengkungan, jumlah titik yang digunakan untuk membagi potongan, dan lain-lain.
- *IcosahedronBufferGeometry*, sebuah kelas untuk mengeneralisasi sebuah geometri *icosahedron*. Konstruktor pada kelas ini menerima parameter berupa radius dengan nilai awal adalah 1 dan detail dengan nilai awal adalah 0.
- *IcosahedronGeometry*, sebuah kelas untuk mengeneralisasi sebuah geometri *icosahedron*. Konstruktor pada kelas ini menerima parameter berupa radius dengan nilai awal adalah 1 dan detail dengan nilai awal adalah 0.
- *LatheBufferGeometry*, merupakan port *BufferGeometry* dari *LatheGeometry*. Konstruktor pada kelas ini menerima parameter berupa *array* dari *Vector2s*, jumlah bagian lingkar yang ingin di generalisasi dengan nilai awal adalah 12, sudut awal dalam radian dengan nilai awal adalah 0, rentang radian dengan nilai awal adalah 2 kali Pi.

```
var points = [];
for ( var i = 0; i < 10; i ++ ) {
        points.push( new THREE.Vector2( Math.sin( i * 0.2 ) * 10 + 5,
        ( i - 5 ) * 2 ) );
}
var geometry = new THREE.LatheBufferGeometry( points );
var material = new THREE.MeshBasicMaterial( { color: 0xffff00 } );
var lathe = new THREE.Mesh( geometry, material );
scene.add( lathe );
```

Listing 2.14: Contoh penggunaan kelas *LatheBufferGeometry*.

- *LatheGeometry*, membuat jala dengan simetri aksial seperti vas. Bentuk ini berotasi di sekitar sumbu Y. Contoh penggunaannya sama seperti kelas *LatheBufferGeometry*.

Konstruktor pada kelas ini menerima parameter berupa *array* dari *Vector2s*, jumlah bagian lingkar yang ingin di generalisasi dengan nilai awal adalah 12, sudut awal dalam radian dengan nilai awal adalah 0, rentang radian dengan nilai awal adalah 2 kali Pi.

- *OctahedronBufferGeometry*, sebuah kelas untuk mengeneralisasi sebuah geometri segi delapan. Konstruktor pada kelas ini menerima parameter berupa radius dengan nilai awal adalah 1 dan detail dengan nilai awal adalah 0.
- *OctahedronGeometry*, sebuah kelas untuk mengeneralisasi sebuah geometri segi delapan. Konstruktor pada kelas ini menerima parameter berupa radius dengan nilai awal adalah 1 dan detail dengan nilai awal adalah 0.
- *ParametricBufferGeometry*, mengeneralisasi geometri yang merepresentasikan permukaan parametrik. Konstruktor pada kelas ini menerima sebuah fungsi yang menerima nilai a dan u di antara 0 sampai dengan 1 dan mengembalikan *Vector3*, banyak potongan, dan banyak tumpukan.

```
var geometry = new THREE.ParametricBufferGeometry(
THREE.ParametricGeometries.klein, 25, 25 );
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
var cube = new THREE.Mesh( geometry, material );
scene.add( cube );
```

Listing 2.15: Contoh penggunaan kelas *ParametricBufferGeometry*.

- *ParametricGeometry*, mengeneralisasi geometri yang merepresentasikan permukaan parametrik. Contoh penggunaannya sama seperti kelas *ParametricBufferGeometry*. Konstruktor pada kelas ini menerima sebuah fungsi yang menerima nilai a dan u di antara 0 sampai dengan 1 dan mengembalikan *Vector3*, banyak potongan, dan banyak tumpukan.
- *PlaneBufferGeometry*, merupakan port *BufferGeometry* dari *PlaneGeometry*. Konstruktor pada kelas ini menerima parameter berupa lebar pada sumbu X dengan nilai awal adalah 1, tinggi pada sumbu Y dengan nilai awal adalah 1, lebar bagian dengan nilai awal adalah 1 dan bersifat fakultatif, dan tinggi bagian dengan nilai awal adalah 1 dan bersifat fakultatif.

```
var geometry = new THREE.PlaneBufferGeometry( 5, 20, 32 );
var material = new THREE.MeshBasicMaterial(
        {color: 0xffff00, side: THREE.DoubleSide}
);
var plane = new THREE.Mesh( geometry, material );
scene.add( plane );
```

Listing 2.16: Contoh penggunaan kelas *PlaneBufferGeometry*.

- *PlaneGeometry*, sebuah kelas untuk mengeneralisasi geometri dataran. Contoh penggunaannya sama seperti kelas *PlaneBufferGeometry*. Konstruktor pada kelas ini menerima parameter berupa lebar pada sumbu X dengan nilai awal adalah 1, tinggi pada sumbu Y dengan nilai awal adalah 1, lebar bagian dengan nilai awal adalah 1 dan bersifat fakultatif, dan tinggi bagian dengan nilai awal adalah 1 dan bersifat fakultatif.
- *PolyhedronBufferGeometry*, merupakan sebuah padat 3 dimensi dengan permukaan datar. Konstruktor pada kelas ini menerima parameter berupa *array* dari titik, *array* dari indeks yang membentuk permukaan, radius dari bentuk akhir, dan detail.

```
var verticesOfCube = [
    -1,-1,-1,    1,-1,-1,    1, 1,-1,    -1, 1,-1,
    -1,-1, 1,    1,-1, 1,    1, 1, 1,    -1, 1, 1,
];
```

```
var indicesOfFaces = [
    2,1,0,     0,3,2,
    0,4,7,     7,3,0,
    0,1,5,     5,4,0,
    1,2,6,     6,5,1,
    2,3,7,     7,6,2,
    4,5,6,     6,7,4
];

var geometry = new THREE.PolyhedronBufferGeometry( verticesOfCube,
 indicesOfFaces, 6, 2 );
```

Listing 2.17: Contoh penggunaan kelas *PolyhedronBufferGeometry*.

- *PolyhedronGeometry*, merupakan sebuah padat 3 dimensi dengan permukaan datar. Konstruktor pada kelas ini menerima parameter berupa *array* dari titik, *array* dari indeks yang membentuk permukaan, radius dari bentuk akhir, dan detail. Contoh penggunaannya sama seperti kelas *PolyhedronBufferGeometry*.
- *RingBufferGeometry*, merupakan port *BufferGeometry* dari *RingGeometry*. Konstruktor pada kelas ini menerima parameter berupa radius bagian dalam dengan nilai awal adalah 20, radius bagian luar dengan nilai awal adalah 50, banyak bagian sudut dengan minimum 3 dan nilai awal 8, banyak bagian Pi dengan minimum 1 dan nilai awal 8, sudut awal dengan nilai awal adalah 0, dan sudut pusat dengan nilai awal adalah dua kali Pi.

```
var geometry = new THREE.RingBufferGeometry( 1, 5, 32 );
var material = new THREE.MeshBasicMaterial(
        { color: 0xffff00, side: THREE.DoubleSide }
);
var mesh = new THREE.Mesh( geometry, material );
scene.add( mesh );
```

Listing 2.18: Contoh penggunaan kelas *RingBufferGeometry*.

- *RingGeometry*, sebuah kelas untuk mengeneralisasi geometri cincin dua dimensi. Konstruktor pada kelas ini menerima parameter berupa radius bagian dalam dengan nilai awal adalah 0.5, radius bagian luar dengan nilai awal adalah 1, banyak bagian sudut dengan minimum 3 dan nilai awal 8, banyak bagian Pi dengan minimum 1 dan nilai awal 8, sudut awal dengan nilai awal adalah 0, dan sudut pusat dengan nilai awal adalah dua kali Pi. Contoh penggunaannya sama seperti kelas *RingBufferGeometry*.
- *ShapeBufferGeometry*, membuat sebuah geometri poligonal satu sisi dari satu atau lebih alur bentuk. Konstruktor pada kelas ini menerima parameter berupa bentuk atau *array* dari bentuk dan jumlah bagian lengkung.

```
var x = 0, y = 0;

var heartShape = new THREE.Shape();

heartShape.moveTo( x + 5, y + 5 );
heartShape.bezierCurveTo( x + 5, y + 5, x + 4, y, x, y );
heartShape.bezierCurveTo( x - 6, y, x - 6, y + 7,x - 6,
 y + 7 );
heartShape.bezierCurveTo( x - 6, y + 11, x - 3, y + 15.4,
 x + 5, y + 19 );
```

```
heartShape.bezierCurveTo( x + 12, y + 15.4, x + 16,
 y + 11, x + 16, y + 7 );
heartShape.bezierCurveTo( x + 16, y + 7, x + 16,
 y, x + 10, y );
heartShape.bezierCurveTo( x + 7, y, x + 5, y + 5,
 x + 5, y + 5 );

var geometry = new THREE.ShapeBufferGeometry( heartShape );
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
var mesh = new THREE.Mesh( geometry, material ) ;
scene.add( mesh );
```

Listing 2.19: Contoh penggunaan kelas *ShapeBufferGeometry*.

- *ShapeGeometry*, membuat sebuah geometri poligonal satu sisi dari satu atau lebih alur bentuk. Konstruktor pada kelas ini menerima parameter berupa bentuk atau *array* dari bentuk dan jumlah bagian lengkung. Contoh penggunaannya sama seperti kelas *ShapeBufferGeometry*.
- *SphereBufferGeometry*, merupakan port *BufferGeometry* dari *SphereGeometry*. Konstruktor pada kelas ini menerima parameter berupa radius dengan nilai awal adalah 50, lebar bagian dengan minimum 3 dan nilai awal adalah 8, tinggi bagian dengan minimum 2 dan nilai awal 6, sudut awal horizontal dengan nilai awal 0, besar sudut horizontal dengan nilai awal adalah dua kali Pi, sudut awal vertikal dengan nilai awal adalah 0, dan besar sudut vertikal dengan nilai awal adalah Pi.

```
var geometry = new THREE.SphereBufferGeometry( 5, 32, 32 );
var material = new THREE.MeshBasicMaterial( {color: 0xffff00} );
var sphere = new THREE.Mesh( geometry, material );
scene.add( sphere );
```

Listing 2.20: Contoh penggunaan kelas *SphereBufferGeometry*.

- *SphereGeometry*, sebuah kelas untuk mengeneralisasi geometri bola. Konstruktor pada kelas ini menerima parameter berupa radius dengan nilai awal adalah 50, lebar bagian dengan minimum 3 dan nilai awal adalah 8, tinggi bagian dengan minimum 2 dan nilai awal 6, sudut awal horizontal dengan nilai awal 0, besar sudut horizontal dengan nilai awal adalah dua kali Pi, sudut awal vertikal dengan nilai awal adalah 0, dan besar sudut vertikal dengan nilai awal adalah Pi. Contoh penggunaannya sama seperti kelas *SphereBufferGeometry*.
- *TetrahedronBufferGeometry*, sebuah kelas untuk mengeneralisasi geometri segi empat. Konstruktor pada kelas ini menerima parameter berupa radius dengan nilai awal adalah 1 dan detail dengan nilai awal adalah 0.
- *TetrahedronGeometry*, sebuah kelas untuk mengeneralisasi geometri segi empat. Konstruktor pada kelas ini menerima parameter berupa radius dengan nilai awal adalah 1 dan detail dengan nilai awal adalah 0.
- *TextBufferGeometry*, sebuah kelas untuk mengeneralisasi tulisan sebagai suatu geometri tunggal. Konstruktor pada kelas ini menerima parameter berupa teks yang ingin ditunjukan dan parameter pendukung lainnya seperti *font*, ukuran, tinggi, dan lain-lain.

```
var loader = new THREE.FontLoader();

loader.load( 'fonts/helvetiker_regular.typeface.json',
function ( font ) {
```

```
        var geometry = new THREE.TextBufferGeometry(
         'Hello three.js!', {
                font: font,
                size: 80,
                height: 5,
                curveSegments: 12,
                bevelEnabled: true,
                bevelThickness: 10,
                bevelSize: 8,
                bevelSegments: 5
        } );
} );
```

Listing 2.21: Contoh penggunaan kelas *TextBufferGeometry*.

- *TextGeometry*, sebuah kelas untuk mengeneralisasi tulisan sebagai suatu geometri tunggal. Konstruktor pada kelas ini menerima parameter berupa teks yang ingin ditunjukan dan parameter pendukung lainnya seperti *font*, ukuran, tinggi, dan lain-lain. Contoh penggunaannya sama seperti kelas *TextBufferGeometry*.
- *TorusBufferGeometry*, merupakan port *BufferGeometry* dari *TorusGeometry*. Konstruktor pada kelas ini menerima parameter berupa radius dengan nilai awal adalah 100, diameter tabung dengan nilai awal adalah 40, banyak bagian radial dengan nilai awal adalah 8, banyak bagian tabung dengan nilai awal adalah 6, sudut pusat dengan nilai awal adalah dua kali Pi.

```
var geometry = new THREE.TorusBufferGeometry( 10, 3, 16, 100 );
var material = new THREE.MeshBasicMaterial( { color: 0xffff00 } );
var torus = new THREE.Mesh( geometry, material );
scene.add( torus );
```

Listing 2.22: Contoh penggunaan kelas *TorusBufferGeometry*.

- *TorusGeometry*, sebuah kelas untuk mengeneralisasi geometri torus. Konstruktor pada kelas ini menerima parameter berupa radius dengan nilai awal adalah 1, diameter tabung dengan nilai awal adalah 0.04, bagian radial dengan nilai awal adalah 8, bagian tabung dengan nilai awal adalah 6, sudut pusat dengan nilai awal adalah dua kali Pi. Contoh penggunaannya sama seperti kelas *TorusBufferGeometry*.
- *TorusKnotBufferGeometry*, merupakan port *BufferGeometry* dari *TorusKnotGeometry*. Konstruktor pada kelas ini menerima parameter berupa radius dengan nilai awal adalah 100, diameter tabung dengan nilai awal adalah 40, banyak bagian tabung dengan nilai awal adalah 64, banyak bagian radial dengan nilai awal adalah 8, jumlah rotasi pada sumbu dengan nilai awal adalah 2, dan jumlah putaran dengan nilai awal adalah 3.

```
var geometry = new THREE.TorusKnotBufferGeometry( 10, 3, 100, 16 );
var material = new THREE.MeshBasicMaterial( { color: 0xffff00 } );
var torusKnot = new THREE.Mesh( geometry, material );
scene.add( torusKnot );
```

Listing 2.23: Contoh penggunaan kelas *TorusKnotBufferGeometry*.

- *TorusKnotGeometry*, membuat simpul knot dengan bagian bentuk yang didefinisikan dengan sepasang bilangan bulat koprima p dan q. Konstruktor pada kelas ini menerima parameter berupa radius dengan nilai awal adalah 100, diameter tabung dengan nilai awal adalah 40, banyak bagian tabung dengan nilai awal adalah 64, banyak bagian radial dengan nilai awal adalah 8, jumlah rotasi pada sumbu dengan nilai awal adalah 2, dan

jumlah putaran dengan nilai awal adalah 3. Contoh penggunaannya sama seperti kelas *TorusKnotBufferGeometry*.

- *TubeGeometry*, membuat sebuah tabung yang diekstrusi sepanjang 3 dimensi melengkung. Konstruktor pada kelas ini menerima parameter berupa alur dengan basis kelas *Curve*, banyak bagian tabung dengan nilai awal 64, radius dengan nilai awal adalah 1, banyak bagian radius dengan nilai awal adalah 8, dan sebuah boolean yang menyatakan tabung tersebut tertutup atau terbuka.

```
function CustomSinCurve( scale ) {

        THREE.Curve.call( this );

        this.scale = ( scale === undefined ) ? 1 : scale;

}

CustomSinCurve.prototype = Object.create( THREE.Curve.prototype );
CustomSinCurve.prototype.constructor = CustomSinCurve;

CustomSinCurve.prototype.getPoint = function ( t ) {

        var tx = t * 3 - 1.5;
        var ty = Math.sin( 2 * Math.PI * t );
        var tz = 0;

        return new THREE.Vector3( tx, ty, tz ).multiplyScalar(
         this.scale );

};

var path = new CustomSinCurve( 10 );
var geometry = new THREE.TubeGeometry( path, 20, 2, 8, false );
var material = new THREE.MeshBasicMaterial( { color: 0x00ff00 } );
var mesh = new THREE.Mesh( geometry, material );
scene.add( mesh );
```

Listing 2.24: Contoh penggunaan kelas *TubeGeometry*.

- *TubeBufferGeometry*, membuat sebuah tabung yang diekstrusi sepanjang 3 dimensi melengkung. Konstruktor pada kelas ini menerima parameter berupa alur dengan basis kelas *Curve*, banyak bagian tabung dengan nilai awal 64, radius dengan nilai awal adalah 1, banyak bagian radius dengan nilai awal adalah 8, dan sebuah boolean yang menyatakan tabung tersebut tertutup atau terbuka. Contoh penggunaannya sama seperti kelas *TubeGeometry*.
- *WireframeGeometry*, dapat digunakan sebagai objek pembantu untuk menampilkan sebuah objek geometri sebagai *wireframe*.

```
var geometry = new THREE.SphereBufferGeometry( 100, 100, 100 );

var wireframe = new THREE.WireframeGeometry( geometry );

var line = new THREE.LineSegments( wireframe );
line.material.depthTest = false;
```

```
line.material.opacity = 0.25;
line.material.transparent = true;

scene.add( line );
```

Listing 2.25: Contoh penggunaan kelas *WireframeGeometry*.

- *Lights*
  - *AmbientLight*, sebuah cahaya yang menyinari objek secara global dan merata. Konstruktor pada kelas ini menerima parameter berupa warna dalam RGB dan intensitas.

```
var light = new THREE.AmbientLight( 0x404040 );
scene.add( light );
```

Listing 2.26: Contoh penggunaan kelas *AmbientLight*.

  - *DirectionalLight*, sebuah pancaran sinar dari arah yang spesifik. Konstruktor pada kelas ini menerima parameter berupa warna dalam heksadesimal dan intensitas. Contoh penggunaannya sama seperti kelas *AmbientLight*.
  - *HemisphereLight*, sebuah cahaya yang penyinaran dilakukan tepat di atas layar dengan peleburan warna langit ke warna lantai. Konstruktor pada kelas ini menerima parameter berupa warna langit dalam heksadesimal , warna daratan dalam heksadesimal, dan intensitas. Contoh penggunaannya sama seperti kelas *AmbientLight*.
  - *Light*, kelas abstrak untuk *Lights*. Konstruktor pada kelas ini menerima parameter berupa warna dalam heksimal dan intensitas.
  - *PointLight*, sebuah pancaran dari satu titik pada setiap arah. Konstruktor pada kelas ini menerima parameter berupa warna dalam heksadesimal, intensitas, jarak dari cahaya saat intensitasnya 0, dan hilangnya cahaya dari pandangan dengan nilai awal adalah 1. Contoh penggunaannya sama seperti kelas *AmbientLight*.
  - *RectAreaLight*, sebuah pancaran sinar seragam melewati permukaan bidang persegi panjang. Konstruktor pada kelas ini menerima parameter berupa warna dalam heksadesimal, intensitas dengan nilai awal adalah 1, lebar cahaya dan tinggi cahaya dengan nilai awal adalah 10.

```
var width = 2;
var height = 10;
var rectLight = new THREE.RectAreaLight(
0xffffff, undefined, width, height );
rectLight.intensity = 70.0;
rectLight.position.set( 5, 5, 0 );
scene.add( rectLight )

rectLightHelper = new THREE.RectAreaLightHelper( rectLight );
scene.add( rectLightHelper );
```

Listing 2.27: Contoh penggunaan kelas *RectAreaLight*.

  - *SpotLight*, sebuah pancaran dari satu titik pada setiap arah sepanjang bidang yang ukurannya dapat bertambah lebih jauh. Konstruktor pada kelas ini menerima parameter berupa warna dalam heksadesimal, intensitas dengan nilai awal adalah 1, jarak maksimal cahaya dari sumber, sudut maksimum, *penumbra*, dan hilangnya cahaya dari pandangan dengan nilai awal adalah 1.

```
var spotLight = new THREE.SpotLight( 0xffffff );
spotLight.position.set( 100, 1000, 100 );

spotLight.castShadow = true;

spotLight.shadow.mapSize.width = 1024;
spotLight.shadow.mapSize.height = 1024;

spotLight.shadow.camera.near = 500;
spotLight.shadow.camera.far = 4000;
spotLight.shadow.camera.fov = 30;

scene.add( spotLight );
```

Listing 2.28: Contoh penggunaan kelas *SpotLight*.

- *Loaders*
  - *AnimationLoader*, kelas untuk memuat animasi dalam format JSON. Konstruktor pada kelas ini menerima parameter berupa *loadingManager*.

```
// instansiasi pemuat
var loader = new THREE.AnimationLoader();

// memuat sumber daya
loader.load(
        // URL sumber daya
        'animations/animation.js',
        // fungsi yang dijalankan saat sumber data telah dimuat
        function ( animation ) {
                // melakukan sesuatu dengan animasi
        },
        // fungsi yang dipanggil saat unduh dalam proses
        function ( xhr ) {
                console.log( (xhr.loaded / xhr.total * 100) + '% loaded' );
        },
        // fungsi yang dipanggil saat unduh gagal
        function ( xhr ) {
                console.log( 'An error happened' );
        }
);
```

Listing 2.29: Contoh penggunaan kelas *AnimationLoader*.

  - *CubeTextureLoader*, kelas untuk memuat sebuah *CubeTexture*. Konstruktor pada kelas ini menerima parameter berupa *loadingManager*.

```
var scene = new THREE.Scene();
scene.background = new THREE.CubeTextureLoader()
        .setPath( 'textures/cubeMaps/' )
        .load( [
                '1.png',
                '2.png',
                '3.png',
```

```
                '4.png',
                '5.png',
                '6.png'
        ] );
```

Listing 2.30: Contoh penggunaan kelas *CubeTextureLoader* menggunakan gambar dengan format PNG di setiap sisinya.

- *DataTextureLoader*, kelas dasar abstrak untuk memuat format tekstur biner umum. Konstruktor pada kelas ini menerima parameter berupa *loadingManager*.
- *FileLoader*, kelas level rendah untuk memuat sumber daya dengan *XMLHTTPRequest*. Kelas ini digunakan secara internal untuk kebanyakan *loaders*. Konstruktor pada kelas ini menerima parameter berupa *loadingManager*.

```
var loader = new THREE.FileLoader();

//memuat sebuah file teks keluaran ke konsol
loader.load(
    // sumber daya URL
    'example.txt',

    // fungsi yang dijalankan saat sumber daya telah dimuat
    function ( data ) {
        // keluaran teks ke konsol
        console.log( data )
    },

    //fungsi yang dipanggil saat unduh dalam proses
    function ( xhr ) {
        console.log( (xhr.loaded / xhr.total * 100) + '% loaded' );
    },

    // fungsi yang dipanggil saat unduh gagal
    function ( xhr ) {
        console.error( 'An error happened' );
    }
);
```

Listing 2.31: Contoh penggunaan kelas *FileLoader* untuk berkas dengan format TXT.

- *FontLoader*, kelas untuk memuat sebuah font dalam format JSON. Konstruktor pada kelas ini menerima parameter berupa *loadingManager*.

```
var loader = new THREE.FontLoader();
var font = loader.load(
        // sumber daya URL
        'fonts/helvetiker_bold.typeface.json'\
        // fungsi yang dijalankan saat sumber daya telah dimuat
        function ( font ) {
                // melakukan sesuatu dengan font
                scene.add( font );
        },
        // fungsi yang dipanggil saat unduh dalam proses
        function ( xhr ) {
```

```
                console.log( (xhr.loaded / xhr.total * 100)
                 + '% loaded' );
        },
        // fungsi yang dipanggil saat unduh gagal
        function ( xhr ) {
                console.log( 'An error happened' );
        }
);
```

Listing 2.32: Contoh penggunaan kelas *FontLoader*.

– *ImageLoader*, sebuah pemuat untuk memuat gambar. Konstruktor pada kelas ini menerima parameter berupa *loadingManager*.

```
// inisiasi pemuat
var loader = new THREE.ImageLoader();

// load a image resource
loader.load(
        // sumber daya URL
        'textures/skyboxsun25degtest.png',
        // fungsi yang dijalankan saat sumber daya telah dimuat
        function ( image ) {
                // melakukan sesuatu dengan gambar

                // menggambar bagian dari gambar pada canvas
                var canvas = document.createElement( 'canvas' );
                var context = canvas.getContext( '2d' );
                context.drawImage( image, 100, 100 );
        },
        // fungsi yang dipanggil saat unduh dalam proses
        function ( xhr ) {
                console.log( (xhr.loaded / xhr.total * 100)
                 + '% loaded' );
        },
        // fungsi yang dipanggil saat unduh gagal
        function ( xhr ) {
                console.log( 'An error happened' );
        }
);
```

Listing 2.33: Contoh penggunaan kelas *ImageLoader*.

– *JSONLoader*, sebuah pemuat untuk memuat objek dalam format JSON. Konstruktor pada kelas ini menerima parameter berupa *loadingManager*.

```
// inisiasi pemuat
var loader = new THREE.JSONLoader();

// memuat sumber daya
loader.load(

        // sumber daya URL
        'models/animated/monster/monster.js',
```

```
        // fungsi yang dijalankan saat sumber daya telah dimuat
        function ( geometry, materials ) {

                var material = materials[ 0 ];
                var object = new THREE.Mesh( geometry, material );

                scene.add( object );

        }
);
```

Listing 2.34: Contoh penggunaan kelas *JSONLoader*.

- *Loader*, kelas dasar untuk implementasi pemuat.
- *MaterialLoader*, sebuah pemuat untuk memuat *Material* dalam format JSON. Konstruktor pada kelas ini menerima parameter berupa *loadingManager*.

```
// inisiasi pemuat
var loader = new THREE.MaterialLoader();

// memuat sumber daya
loader.load(
        // sumber daya URL
        'path/to/material.json',
        // fungsi yang dijalankan saat sumber daya telah dimuat
        function ( material ) {
                object.material = material;
        },
        // fungsi yang dipanggil saat unduh dalam proses
        function ( xhr ) {
                console.log( (xhr.loaded / xhr.total * 100)
                 + '% loaded' );
        },
        // fungsi yang dipanggil saat unduh gagal
        function ( xhr ) {
                console.log( 'An error happened' );
        }
);
```

Listing 2.35: Contoh penggunaan kelas *MaterialLoader*.

- *ObjectLoader*, sebuah pemuat untuk memuat sumber daya JSON. Konstruktor pada kelas ini menerima parameter berupa *loadingManager*.

```
var loader = new THREE.ObjectLoader();

loader.load(
    // sumber daya URL
    "models/json/example.json",

    // mengirimkan data yang telah dimuat ke fungsi onLoad
    // di sini diasumsikan mejadi sebuah objek
    function ( obj ) {
                // menambahkan objek yang telah dimuat ke layar
```

```
        scene.add( obj );
    },

    // fungsi yang dipanggil saat unduh dalam proses
    function ( xhr ) {
        console.log( (xhr.loaded / xhr.total * 100)
         + '% loaded' );
    },

    // fungsi yang dipanggil saat unduh gagal
    function ( xhr ) {
        console.error( 'An error happened' );
    }
);


// sebagai alternatif untuk mengurai JSON yang telah dimuat
var object = loader.parse( a_json_object );

scene.add( object );
```

Listing 2.36: Contoh penggunaan kelas *ObjectLoader*.

– *TextureLoader*, kelas untuk memuat tekstur. Konstruktor pada kelas ini menerima parameter berupa *loadingManager*.

```
// inisiasi pemuat
var loader = new THREE.TextureLoader();

// memuat sumber daya
loader.load(
        // sumber daya URL
        'textures/land_ocean_ice_cloud_2048.jpg',
        // fungsi yang dijalankan saat sumber daya telah dimuat
        function ( texture ) {
                // melakukan sesuatu dengan tekstur
                var material = new THREE.MeshBasicMaterial( {
                        map: texture
                 } );
        },
        // fungsi yang dipanggil saat unduh dalam proses
        function ( xhr ) {
                console.log( (xhr.loaded / xhr.total * 100)
                 + '% loaded' );
        },
        // fungsi yang dipanggil saat unduh gagal
        function ( xhr ) {
                console.log( 'An error happened' );
        }
);
```

Listing 2.37: Contoh penggunaan kelas *TextureLoader*.

– *MTLLoader*, sebuah pemuat untuk memuat sumber daya .mtl. Pemuat ini digunakan

secara internal pada *OBJMTLLoader* dan *UTS8Loader*. Konstruktor pada kelas ini menerima parameter berupa *loadingManager*.

– *OBJLoader*, sebuah pemuat untuk memuat sumber daya .obj. Konstruktor pada kelas ini menerima parameter berupa *loadingManager*.

```
// inisiasi pemuat
var loader = new THREE.OBJLoader();

// memuat sumber daya
loader.load(
        // sumber daya URL
        'models/monster.obj',
        // fungsi yang dipanggil saat sumber daya telah dimuat
        function ( object ) {
                scene.add( object );
        }
);
```

Listing 2.38: Contoh penggunaan kelas *OBJLoader*.

- *Materials*

  – *LineBasicMaterial*, sebuah bahan untuk menggambar geometri gaya *wireframe*. Konstruktor pada kelas ini menerima parameter berupa objek dan bersifat fakultatif.

```
var material = new THREE.LineBasicMaterial( {
        color: 0xffffff,
        linewidth: 1,
        linecap: 'round', //ignored by WebGLRenderer
        linejoin:  'round' //ignored by WebGLRenderer
} );
```

Listing 2.39: Contoh penggunaan kelas *LineBasicMaterial*.

  – *LineDashedMaterial*, sebuah bahan untuk menggambar geometri gaya *wireframe* dengan garis putus-putus. Konstruktor pada kelas ini menerima parameter berupa objek dan bersifat fakultatif.

```
var material = new THREE.LineDashedMaterial( {
        color: 0xffffff,
        linewidth: 1,
        scale: 1,
        dashSize: 3,
        gapSize: 1,
} );
```

Listing 2.40: Contoh penggunaan kelas *LineDashMaterial*.

  – *Material*, kelas dasar abstrak untuk bahan.

  – *MeshBasicMaterial*, sebuah bahan untuk menggambar geometri dengan cara sederhana yang datar. Konstruktor pada kelas ini menerima parameter berupa objek dan bersifat fakultatif.

  – *MeshDepthMaterial*, sebuah bahan untuk menggambar geometri berdasarkan kedalaman. Konstruktor pada kelas ini menerima parameter berupa objek dan bersifat fakultatif.

- *MeshLambertMaterial*, sebuah bahan untuk permukaan yang tidak bercahaya. Konstruktor pada kelas ini menerima parameter berupa objek dan bersifat fakultatif.
- *MeshNormalMaterial*, sebuah bahan yang memetakan vektor normal ke warna RGB. Konstruktor pada kelas ini menerima parameter berupa objek dan bersifat fakultatif.
- *MeshPhongMaterial*, sebuah bahan untuk permukaan yang bercahaya dengan sorotan cahaya. Konstruktor pada kelas ini menerima parameter berupa objek dan bersifat fakultatif.
- *MeshPhysicalMaterial*, sebuah ekstensi dari *MeshStandardMaterial* yang memungkinkan kontrol yang lebih kuat terhadap daya pemantulan. Konstruktor pada kelas ini menerima parameter berupa objek dan bersifat fakultatif.
- *MeshStandardMaterial*, sebuah fisik bahan dasar standar menggunakan alur kerja *Metallic-Roughness*. Konstruktor pada kelas ini menerima parameter berupa objek dan bersifat fakultatif.
- *MeshToonMaterial*, sebuah ekstensi dari *MeshPhongMaterial* dengan bayangan. Konstruktor pada kelas ini menerima parameter berupa objek dan bersifat fakultatif.
- *PointsMaterial*, sebuah bahan dasar yang digunakan *Points*. Konstruktor pada kelas ini menerima parameter berupa objek dan bersifat fakultatif.

```
var starsGeometry = new THREE.Geometry();

for ( var i = 0; i < 10000; i ++ ) {

        var star = new THREE.Vector3();
        star.x = THREE.Math.randFloatSpread( 2000 );
        star.y = THREE.Math.randFloatSpread( 2000 );
        star.z = THREE.Math.randFloatSpread( 2000 );

        starsGeometry.vertices.push( star );

}

var starsMaterial = new THREE.PointsMaterial( { color: 0x888888 } );

var starField = new THREE.Points( starsGeometry, starsMaterial );

scene.add( starField );
```

Listing 2.41: Contoh penggunaan kelas *PointsMaterial*.

- *RawShaderMaterial*, kelas ini bekerja seperti *ShaderMaterial* kecuali definisi dari *uniform* dan atribut yang telah ada tidak ditambahkan secara otomatis ke GLSL *shader* kode. Konstruktor pada kelas ini menerima parameter berupa objek dan bersifat fakultatif.

```
var material = new THREE.RawShaderMaterial( {

    uniforms: {
        time: { value: 1.0 }
    },
    vertexShader: document.getElementById( 'vertexShader' )
    .textContent,
    fragmentShader: document.getElementById( 'fragmentShader')
    .textContent,
```

```
} );
```

Listing 2.42: Contoh penggunaan kelas *RawShaderMaterial*.

– *ShaderMaterial*, sebuah bahan yang dibangun dengan *shader* kustom. Konstruktor pada kelas ini menerima parameter berupa objek dan bersifat fakultatif.

```
var material = new THREE.ShaderMaterial( {

        uniforms: {

                time: { value: 1.0 },
                resolution: { value: new THREE.Vector2() }

        },

        vertexShader: document.getElementById( 'vertexShader' )
        .textContent,

        fragmentShader: document.getElementById( 'fragmentShader')
        .textContent

} );
```

Listing 2.43: Contoh penggunaan kelas *ShaderMaterial*.

– *ShadowMaterial*, sebuah bahan yang dapat menerima bayangan tetapi jika tidak merima bayangan maka akan transparan. Konstruktor pada kelas ini menerima parameter berupa objek dan bersifat fakultatif.

```
var planeGeometry = new THREE.PlaneGeometry( 2000, 2000 );
planeGeometry.rotateX( - Math.PI / 2 );

var planeMaterial = new THREE.ShadowMaterial();
planeMaterial.opacity = 0.2;

var plane = new THREE.Mesh( planeGeometry, planeMaterial );
plane.position.y = -200;
plane.receiveShadow = true;
scene.add( plane );
```

Listing 2.44: Contoh penggunaan kelas *ShadowMaterial*.

– *SpriteMaterial*, sebuah bahan yang digunakan dengan *Sprite*. Konstruktor pada kelas ini menerima parameter berupa objek dan bersifat fakultatif.

```
var spriteMap = new THREE.TextureLoader().load( 'textures/sprite.png' );

var spriteMaterial = new THREE.SpriteMaterial( {
 map: spriteMap, color: 0xffffff } );

var sprite = new THREE.Sprite( spriteMaterial );
sprite.scale.set(200, 200, 1)

scene.add( sprite );
```

Listing 2.45: Contoh penggunaan kelas *SpriteMaterial*.

- *Objects*
  - *Bone*, sebuah tulang yang merupakan bagian dari kerangka.

```
var root = new THREE.Bone();
var child = new THREE.Bone();

root.add( child );
child.position.y = 5;
```

Listing 2.46: Contoh penggunaan kelas *Bone*.

  - *Group*, hampir sama dengan suatu *Object3D*.

```
var geometry = new THREE.BoxBufferGeometry( 1, 1, 1 );
var material = new THREE.MeshBasicMaterial( {color: 0x00ff00} );

var cubeA = new THREE.Mesh( geometry, material );
cubeA.position.set( 100, 100, 0 );

var cubeB = new THREE.Mesh( geometry, material );
cubeB.position.set( -100, -100, 0 );

//create a group and add the two cubes
//These cubes can now be rotated / scaled etc as a group
var group = new THREE.Group();
group.add( cubeA );
group.add( cubeB );

scene.add( group );
```

Listing 2.47: Contoh penggunaan kelas *Group*.

  - *LensFlare*, membuat lensa suar tiruan yang mengikuti cahaya. Konstruktor pada kelas ini menerima parameter berupa tekstur, ukuran, jarak, mode pencampuran, dan warna.

```
var light = new THREE.PointLight( 0xffffff, 1.5, 2000 );

var textureLoader = new THREE.TextureLoader();

var textureFlare = textureLoader.
load( "textures/lensflare/lensflare.png" );

var flareColor = new THREE.Color( 0xffffff );
flareColor.setHSL( h, s, l + 0.5 );

var lensFlare = new THREE.LensFlare( textureFlare,
 700, 0.0, THREE.AdditiveBlending, flareColor );
lensFlare.position.copy( light.position );

scene.add( lensFlare );
```

Listing 2.48: Contoh penggunaan kelas *LensFlare*.

  - *Line*, sebuah garis yang kontinu. Konstruktor pada kelas ini menerima parameter berupa geometri dan material.

```
var material = new THREE.LineBasicMaterial({
        color: 0x0000ff
});

var geometry = new THREE.Geometry();
geometry.vertices.push(
        new THREE.Vector3( -10, 0, 0 ),
        new THREE.Vector3( 0, 10, 0 ),
        new THREE.Vector3( 10, 0, 0 )
);

var line = new THREE.Line( geometry, material );
scene.add( line );
```

Listing 2.49: Contoh penggunaan kelas *Line*.

- *LineLoop*, sebuah line kontinu yang kembali ke awal. Konstruktor pada kelas ini menerima parameter berupa geometri dan material.
- *LineSegments*, beberapa garis yang ditarik antara beberapa pasang *vertex*. Konstruktor pada kelas ini menerima parameter berupa geometri dan material.
- *Mesh*, sebuah kelas yang merepresentasikan object dengan dasar segitiga. Konstruktor pada kelas ini menerima parameter berupa geometri dan material.

```
var geometry = new THREE.BoxBufferGeometry( 1, 1, 1 );
var material = new THREE.MeshBasicMaterial( { color: 0xffff00 } );
var mesh = new THREE.Mesh( geometry, material );
scene.add( mesh );
```

Listing 2.50: Contoh penggunaan kelas *Mesh*.

- *Points*, sebuah kelas yang merepresentasikan titik. Konstruktor pada kelas ini menerima parameter berupa geometri dan material.
- *Skeleton*, sebuah *array* dari tulang untuk membuat kerangka yang bisa digunakan pada *SkinnedMesh*. Konstruktor pada kelas ini menerima parameter berupa *array* dari *bones* dan *array* invers dari *Matriks4s*.

```
var bones = [];

var shoulder = new THREE.Bone();
var elbow = new THREE.Bone();
var hand = new THREE.Bone();

shoulder.add( elbow );
elbow.add( hand );

bones.push( shoulder );
bones.push( elbow );
bones.push( hand );

shoulder.position.y = -5;
elbow.position.y = 0;
hand.position.y = 5;
```

```
var armSkeleton = new THREE.Skeleton( bones );
```

Listing 2.51: Contoh penggunaan kelas *Skeleton*.

– *SkinnedMesh*, sebuah *mesh* yang mempunyai kerangka yang terdiri dari tulang dan digunakan untuk menganimasikan kumpulan *vertex* pada geometri. Konstruktor pada kelas ini menerima parameter berupa geometri dan material.

```
var geometry = new THREE.CylinderBufferGeometry(
5, 5, 5, 5, 15, 5, 30 );

// membuat index kulit dan berat kulit
for ( var i = 0; i < geometry.vertices.length; i ++ ) {

        // fungsi imajiner untuk menghitung index dan berat
        //bagian ini harus diganti bergantung pada kerangka dan model
        var skinIndex = calculateSkinIndex(
        geometry.vertices, i );
        var skinWeight = calculateSkinWeight(
        geometry.vertices, i );

        // menggerakan antara tulang
        geometry.skinIndices.push( new THREE.Vector4(
        skinIndex, skinIndex + 1, 0, 0 ) );
        geometry.skinWeights.push( new THREE.Vector4(
        1 - skinWeight, skinWeight, 0, 0 ) );

}

var mesh = THREE.SkinnedMesh( geometry, material );

// lihat contoh dari THREE.Skeleton untuk armSkeleton
var rootBone = armSkeleton.bones[ 0 ];
mesh.add( rootBone );

// ikat kerangka dengan jala
mesh.bind( armSkeleton );

// pindahkan tulang dan manipulasi model
armSkeleton.bones[ 0 ].rotation.x = -0.1;
armSkeleton.bones[ 1 ].rotation.x = 0.2;
```

Listing 2.52: Contoh penggunaan kelas *SkinnedMesh*.

– *Sprite*, sebuah dataran yang selalu menghadap kamera secara umum dengan bagian tekstur transparan diaplikasikan. Konstruktor pada kelas ini menerima parameter berupa material.

```
var spriteMap = new THREE.TextureLoader().load( "sprite.png" );
var spriteMaterial = new THREE.SpriteMaterial(
{ map: spriteMap, color: 0xffffff } );
var sprite = new THREE.Sprite( spriteMaterial );
scene.add( sprite );
```

Listing 2.53: Contoh penggunaan kelas *Sprite*.

- *Renderers*
- *WebGLRenderer*, pembangun WebGL menampilkan layar indah yang dbuat oleh Anda menggunakan WebGL. Konstruktor pada kelas ini menerima parameter berupa *canvas*, konteks, presisi, dan parameter relevan lainnya.
- *WebGLRenderTarget*, merupakan sebuah penyangga target pembangun yang memungkinkan kartu video menggambarkan piksel untuk layar yang dibangun di latar. Konstruktor pada kelas ini menerima parameter berupa lebar, tinggi, dan parameter relevan lainnya.
- *WebGLRenderTargetCube*, digunakan oleh *CubeCamera* sebagai *WebGLRenderTarget*. Konstruktor pada kelas ini menerima parameter berupa lebar, tinggi, dan parameter relevan lainnya.
- *Scenes*
  - *Fog*, kelas yang berisi parameter untuk mendefinisikan kabut. Konstruktor pada kelas ini menerima parameter berupa warna dalam heksadesimal, jarak terdekat, dan jarak terjauh.
  - it FogExp2, kelas ini berisi parameter pendefinisikan eksponensial kabut yang bertumbuh secara padat eksponensial dengan jarak. Konstruktor pada kelas ini menerima parameter berupa warna dalam heksadesimal dan kecepatan kabut.
  - *Scene*, sebuah layar yang memungkinkan untuk membuat dan menempatkan sesuatu pada pustaka Three.js.
- *Texture*
  - *CanvasTexture*, membuat tekstur dari suatu elemen *canvas*. Konstruktor pada kelas ini menerima parameter berupa *canvas*, *mapping*, wrapS dan wrapT berdasarkan *THREE.ClampToEdgeWrapping*, penyaring besar, penyaring kecil, konstanta, format, tipe, dan *anisotropy*.
  - *CompressedTexture*, membuat tekstur berdasarkan data bentuk kompres. Contohnya dari sebuah berkas DDS. Konstruktor pada kelas ini menerima parameter berupa objek dengan data, lebar, tinggi, format, tipe, *mapping*, wrapS dan wrapT berdasarkan *THREE.ClampToEdgeWrapping*, penyaring besar, penyaring kecil, dan *anisotropy*.
  - *CubeTexture*, membuat tekstur kubus dari 6 buah gambar. Konstruktor pada kelas ini menerima parameter berupa gambar, *mapping*, wrapS dan wrapT berdasarkan *THREE.ClampToEdgeWrapping*, penyaring besar, penyaring kecil, format, tipe, dan *anisotropy*.

```
var loader = new THREE.CubeTextureLoader();
loader.setPath( 'textures/cube/pisa/' );

var textureCube = loader.load( [
        'px.png', 'nx.png',
        'py.png', 'ny.png',
        'pz.png', 'nz.png'
] );

var material = new THREE.MeshBasicMaterial( {
color: 0xffffff, envMap: textureCube
} );
```

Listing 2.54: Contoh penggunaan kelas *CubeTexture*.

- *DataTexture*, membuat tekstur langsung dari data mentah, lebar, dan panjang. Konstruktor pada kelas ini menerima parameter berupa data, lebar, tinggi, format, tipe, *mapping*, wrapS dan wrapT berdasarkan *THREE.ClampToEdgeWrapping*, penyaring besar, penyaring kecil, *anisotropy*, dan format.
- *DepthTexture*, membuat tekstur untuk digunakan sebagai *Depth Texture*. Konstruktor pada kelas ini menerima parameter berupa lebar, tinggi, format, tipe, wrapS dan wrapT berdasarkan *THREE.ClampToEdgeWrapping*, penyaring besar, penyaring kecil, dan *anisotropy*.
- *Texture*, membuat tekstur untuk mengaplikasikan permukaan atau sebagai refleksi. Konstruktor pada kelas ini menerima parameter berupa gambar, *mapping*, wrapS dan wrapT berdasarkan *THREE.ClampToEdgeWrapping*, penyaring besar, penyaring kecil, format, tipe, dan *anisotropy*.

```
var texture = new THREE.TextureLoader().load( "textures/water.jpg" );
texture.wrapS = THREE.RepeatWrapping;
texture.wrapT = THREE.RepeatWrapping;
texture.repeat.set( 4, 4 );
```

Listing 2.55: Contoh penggunaan kelas *Texture*.

- *VideoTexture*, membuat tekstur untuk digunakan sebagai tekstur video. Konstruktor pada kelas ini menerima parameter berupa video, *mapping*, wrapS dan wrapT berdasarkan *THREE.ClampToEdgeWrapping*, penyaring besar, penyaring kecil, format, tipe, dan *anisotropy*.

```
var video = document.getElementById( 'video' );

var texture = new THREE.VideoTexture( video );
texture.minFilter = THREE.LinearFilter;
texture.magFilter = THREE.LinearFilter;
texture.format = THREE.RGBFormat;
```

# LAMPIRAN A

# KODE PROGRAM

Listing A.1: MyCode.c

```c

// This does not make algorithmic sense,
// but it shows off significant programming characters.

#include<stdio.h>

void myFunction( int input, float* output ) {
    switch ( array[i] ) {
        case 1: // This is silly code
            if ( a >= 0 || b <= 3 && c != x )
                *output += 0.005 + 20050;
            char = 'g';
            b = 2^n + ~right_size - leftSize * MAX_SIZE;
            c = (--aaa + &daa) / (bbb++ - ccc % 2 );
            strcpy(a,"hello $@?");
    }
    count = ~mask | 0x00FF00AA;
}

// Fonts for Displaying Program Code in LATEX
// Adrian P. Robson, nepsweb.co.uk
// 8 October 2012
// http://nepsweb.co.uk/docs/progfonts.pdf
```

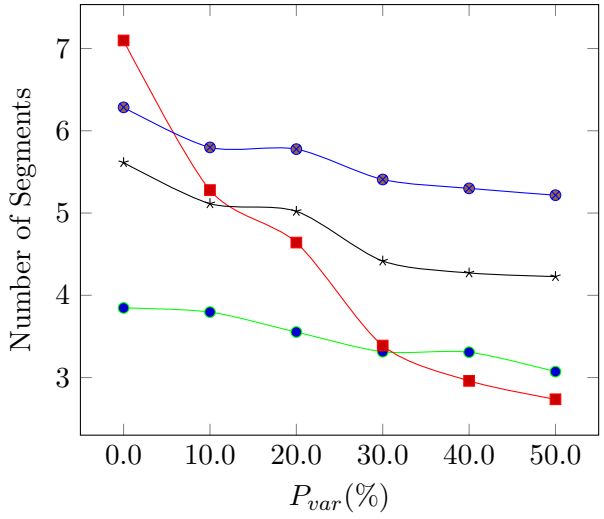
Listing A.2: MyCode.java

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                   //id of the set
    protected MyEdge FurthestEdge;                      //the furthest edge
    protected HashSet<MyVertex> set;                    //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;    //list of all vertices in the set for each trajectory
    protected ArrayList<Integer> closeID;               //store the ID of all vertices
    protected ArrayList<Double> closeDist;              //store the distance of all vertices
    protected int totaltrj;                             //total trajectories in the set

    /*
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

}
```

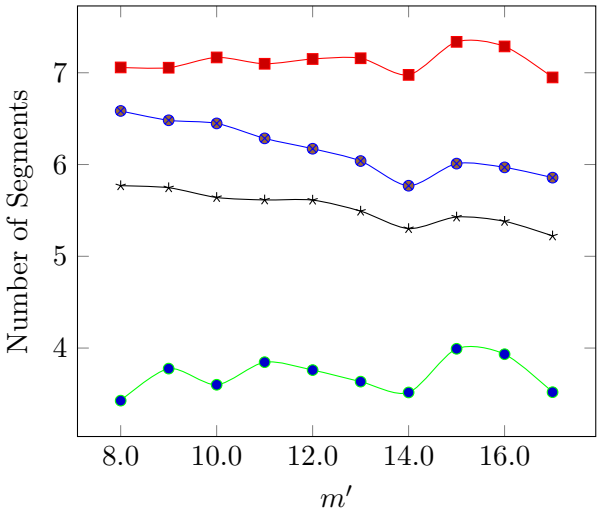
# LAMPIRAN B

# HASIL EKSPERIMEN

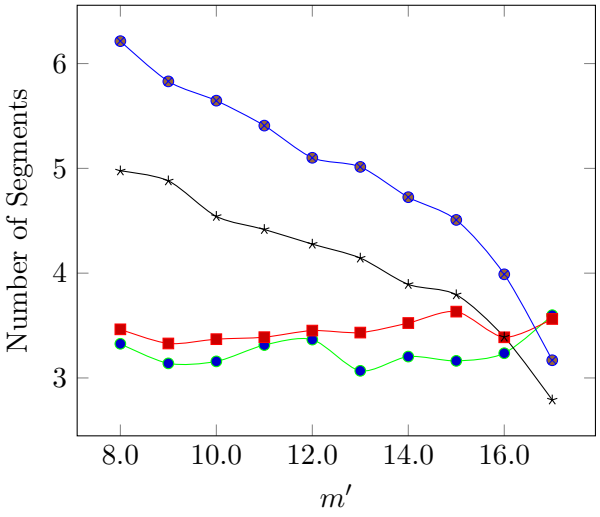
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



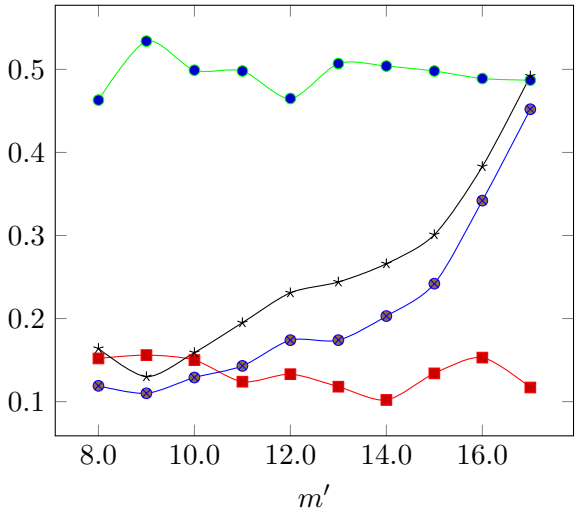

Gambar B.1: Hasil 1




Gambar B.2: Hasil 2




Gambar B.3: Hasil 3




Gambar B.4: Hasil 4