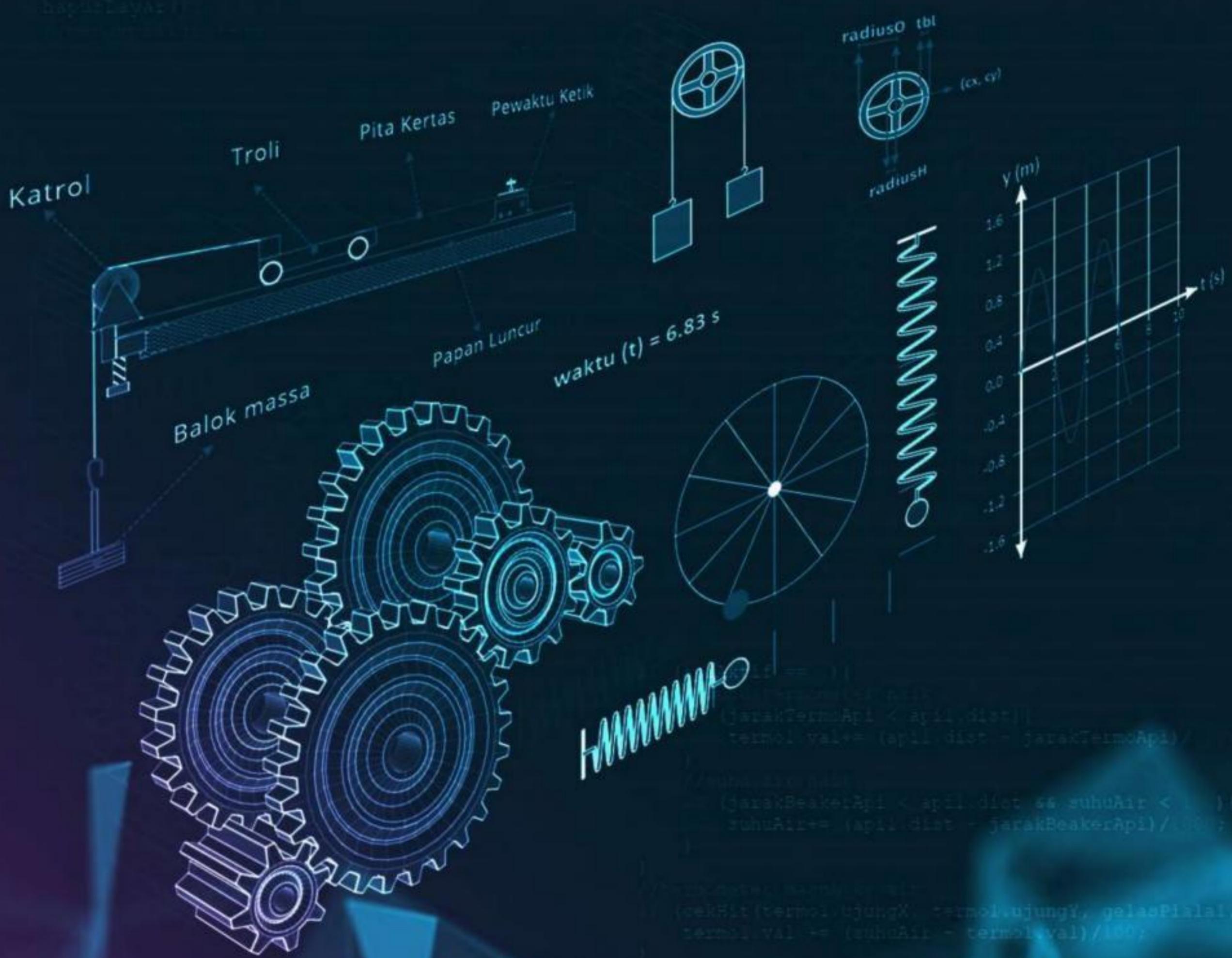


LABORATORIUM VIRTUAL

KONSEP DAN PENGEMBANGAN SIMULASI FISIKA



LABORATORIUM VIRTUAL

KONSEP DAN PENGEMBANGAN SIMULASI FISIKA

Wandah Wibawanto

Penerbit LPPM UNNES

**LABORATORIUM VIRTUAL
KONSEP DAN PENGEMBANGAN SIMULASI FISIKA**
● All Rights Reserved
Hak cipta dilindungi undang-undang

Penulis:
Wandah Wibawanto, S.Sn. M.Ds.

Editor (verifikator ahli):
**Prof. Dr. rer.nat. Wahyu Hardyanto, M. Si.
Fianti, S. Si., M. Sc., Ph. D**

Desain & Tata Letak:
Wandah Wibawanto, S.Sn. M.Ds.

Penerbit:
Penerbit LPPM UNNES
Gedung Prof. Retno Sriningsih Satmoko Lt 1
Universitas Negeri Semarang
Kampus Sekaran, Gunungpati, Semarang
e-mail: lp2m@mail.unnes.ac.id

ISBN : 978-623-7618-84-3

Cetakan Pertama : 2020

**Undang - Undang RI Nomor 19 Tahun 2002
Tentang Hak Cipta**

Ketentuan Pidana
Pasal 72 (ayat 2)
Barang Siapa dengan sengaja menyiaran, memamerkan, mengedarkan, atau menjual kepada umum suatu ciptaan atau barang hasil pelanggaran Hak Cipta atau hak terkait sebagaimana dimaksud pada ayat (1), dipidana dengan pidana penjara paling lama 5 (lima) tahun dan/atau denda paling banyak Rp. 500.000.000,00 (lima ratus juta rupiah).

Untuk ilmuwan masa depan
Azka SaturnLab & Izzan Artwork



KATA PENGANTAR

Alhamdulillahi robbil'aalamin, pada akhirnya buku ini selesai ditulis setelah proses penulisan dan pengembangan aplikasi berjalan sejak awal 2020. Rasa syukur yang terdalam saya sampaikan karena diberikan kemudahan dalam penulisan buku ini.

Istilah laboratorium virtual merupakan istilah yang menjadi *hype* dalam 1 dekade terakhir. Keberadaan teknologi seperti *augmented reality* dan *virtual reality* menjadikan banyak pengembang aplikasi merilis laboratorium virtual dalam berbagai format, mulai dari *platform* umum seperti *mobilephone* dan website sampai dengan aplikasi yang membutuhkan perangkat khusus seperti *Head Mounted Display* untuk mensimulasikan laboratorium virtual dalam bentuk realitas maya.

Sementara itu, pengembangan laboratorium virtual oleh pengembang lokal atau pengajar tanah air, masih relatif minim. Hal ini mungkin disebabkan karena minimnya referensi, atau sudah merasa nyaman dengan penggunaan aplikasi yang telah dibuat oleh pengembang luar negeri. Untuk itu, terbesit sekilas dipikiran saya untuk menuliskan sedikit referensi tentang konsep dan pengembangan laboratorium virtual dengan tujuan untuk menumbuhkan ketertarikan kepada siapapun dalam mengembangkan media pembelajaran sains berupa laboratorium virtual, yang mungkin dapat digunakan dalam ruang lingkup kecil yaitu pembelajaran di kelas atau bahkan dikembangkan lebih lanjut untuk menjadi aplikasi yang digunakan secara global.

Buku ini saya tulis dengan latar belakang saya yang bukan dari bidang fisika, dan lebih cenderung kepada teknis pemrograman berbasis HTML 5 dan Javascript. Adapun ketika pembaca dari bidang Fisika atau sains lainnya menemukan beberapa kesalahan konsep, maka saya memohon maaf yang sebesar-besarnya dan memohon masukan guna perbaikan selanjutnya. Akhirnya, semoga buku ini dapat membawa manfaat bagi pengembangan aplikasi laboratorium virtual di Indonesia.

Semarang, Juni 2020
Wandah Wibawanto

Daftar Isi

Kata Pengantar	v
Daftar Isi	vii
Daftar Gambar	xi
BAB 1 Laboratorium Virtual	1
1.1 Konsep Dasar Laboratorium Virtual.....	1
1.2 Tujuan dan Keunggulan Penggunaan Laboratorium Virtual	5
1.3 Terminologi dalam Laboratorium Virtual	11
1.4 Mendesain dan Mengembangkan Laboratorium Virtual.....	14
A. Metodologi pengembangan laboratorium virtual.....	14
B. Pengetahuan dasar dalam pengembangan laboratorium virtual.....	16
1.5 Pengembangan Laboratorium Virtual dalam Buku ini	17
1.6 Referensi Pengembangan Laboratorium Virtual.....	18
1.7 Lisensi Buku.....	18
BAB 2 Teknologi HTML 5 dan Canvas	19
2.1 HTML Canvas	19
2.2 Struktur HTML Canvas	21
2.3 Javascript dan Kode untuk Canvas	22
A. Tutorial 1 – Memahami Javascript dan Canvas	22
2.4 Debugging (Menguji Kode / Mencari Kesalahan Kode).....	30
BAB 3 Pengembangan Simulasi dengan Javascript	33
3.1 Konsep Pengembangan Simulasi.....	33
3.2 Tahapan Perencanaan	35
3.3 Tahapan Pengembangan Antarmuka (<i>Graphic Modelling</i>).....	36
A. Library Javascript.....	37
3.4 Tahapan Pengembangan Interaktivitas (<i>Interactive Construction</i>).....	38
3.5 Tahapan Distribusi (<i>Deployment</i>)	39
3.6 Sumber Belajar Javascript.....	39

BAB 4 Grafis Dasar	41
4.1 Menggunakan <i>File Library</i> simulasiDasar.js.....	41
4.2 Struktur utama <i>file</i> HTML	42
4.3 Membuat grafis dasar	42
A. Tutorial 2a – Membuat Garis.....	42
B. Tutorial 2b – Membuat Kotak.....	46
C. Membuat Lingkaran.....	47
D. Membuat Diagram Kartesius (Sistem Kordinat Kartesius).....	48
E. Menggabungkan Kode <i>Library</i> dengan Kode JS.....	50
4.4 Transformasi Sistem Koordinat Layar.....	53
A. Tutorial 2c – Membuat Garis pada Sistem Koordinat Kartesian.....	55
 BAB 5 Operasi Teks dengan Javascript	57
5.1 Fungsi <i>fillText</i> pada canvas.....	57
5.2 Fungsi teks pada simulasiDasar.js	57
A. Tutorial 3a – Operasi Teks	57
B. Tutorial 3b – <i>teksHTML</i>	59
C. Tutorial 3c – Membuat Input Teks	60
5.3 Formula Matematika dengan <i>Latex Equation</i>	65
 BAB 6 Interaktivitas	67
6.1 Membuat tombol pada canvas.....	67
A. Tutorial 4a – Membuat Tombol.....	67
B. Tutorial 4b – <i>Slider</i>	70
C. Tutorial 5 – <i>Drag and Drop</i>	73
 BAB 7 Animasi	81
7.1 Konsep Dasar Animasi dengan Canvas.....	81
A. Tutorial 6 – Animasi Roda Gigi.....	82
7.2 Animasi <i>Sprite</i>	85
A. Tutorial 7 – Animasi <i>Sprite</i>	86

BAB 8 Penjumlahan Vektor	89
8.1 Menentukan Penjumlahan Vektor	89
A. Tutorial 8 – Penjumlahan Vektor	90
BAB 9 Simulasi Gerak	95
9.1 Gerak Lurus.....	95
A. Tutorial 9 – Kecepatan antara 2 Objek.....	95
B. Tutorial 10 – Simulasi Gerak Lurus Berubah Beraturan	104
9.2 Gerak Parabola.....	110
A. Tutorial 11 – Gerak Parabola	111
9.3 Gerak Melingkar	117
A. Tutorial 12 – Gerak Melingkar	117
BAB 10 Simulasi Gaya	125
10.1 Simulasi Hukum Newton	125
A. Tutorial 13 – Hukum Newton tentang Gerak.....	126
B. Tutorial 14 – Mesin Atwood	131
C. Tutorial 15 – Sistem Katrol Ganda	137
BAB 11 Simulasi Suhu	143
11.1 Simulasi Konversi Suhu.....	143
A. Tutorial 16 – Konversi Suhu.....	143
11.2 Simulasi Pengukuran Panas	147
A. Tutorial 17 – Pengukuran Panas.....	147
BAB 12 Simulasi Tambahan	147
12.1 Simulasi Lensa	147
A. Tutorial 18 – Simulasi Lensa	147
12.2 Simulasi Medan Listrik	152
A. Tutorial 19 – Simulasi Medan Listrik.....	152
BAB 13 GUI (<i>Graphic User Interface</i>)	157
13.1 GUI (Antar Muka)	157
13.2 Menyiapkan Simulasi yang Akan Ditampilkan dalam Aplikasi.....	159
A. Tutorial 20 – GUI Virtual Lab	160

BAB 14 Penutup	171
14.1 Langkah Selanjutnya.....	171
14.2 Meningkatkan Fitur Grafis 3 Dimensi	173
14.3 Saran terkait Pengembangan Laboratorium Virtual.....	174
Daftar Pustaka	175
Tentang Penulis	177

Daftar Gambar

Gambar 1.1 Laboratorium Virtual PhET (<i>low immersion</i>)	3
Gambar 1.2 Simulasi <i>International Space Station ISS NASA</i>	4
Gambar 1.3 Laboratorium Virtual Labster.....	5
Gambar 1.4 Zona <i>Goldilock</i>	6
Gambar 1.5 Mikroskop confocal virtual.....	7
Gambar 1.6 Visualisasi sel dalam format 3 dimensi pada lab virtual.....	8
Gambar 1.7 Fitur kontrol waktu pada laboratorium virtual	9
Gambar 1.8 Kesalahan praktikum dalam laboratorium virtual	10
Gambar 1.9 Gamifikasi pada SuperChem VR.....	11
Gambar 1.10 AR Tata Surya menggunakan <i>marker</i> (penanda).....	12
Gambar 1.11 Penggunaan HMD untuk menjalankan aplikasi laboratorium virtual.....	13
Gambar 1.12 Penggunaan <i>Mixed Reality</i>	14
Gambar 1.13 Metode pengembangan laboratorium virtual	15
Gambar 2.1 Penulisan kode HTML dan JS di Notepad++.....	23
Gambar 2.2 Hasil tutorial 1, gambar garis pada canvas.....	23
Gambar 2.3 Metode menggambar garis pada canvas.....	24
Gambar 2.4 Kesalahan yang dimunculkan pada panel <i>console</i>	31
Gambar 2.5 Luaran pada panel <i>console</i>	31
Gambar 3.1 Pemodelan Matematika	33
Gambar 3.2 Praktikum gerak lurus berubah beraturan	35
Gambar 3.3 Skema praktikum gerak lurus berubah beraturan	36
Gambar 3.4 Antarmuka percobaan gerak lurus berubah beraturan	37
Gambar 3.5 Elemen interaktif pada simulasi gerak parabola.....	39
Gambar 4.1 Struktur <i>file</i> dan <i>folder</i> proyek simulasi.....	41
Gambar 4.2 Hasil tutorial membuat garis	44
Gambar 4.3 Judul pada panel internet <i>browser</i>	44
Gambar 4.4 Menentukan kode warna dengan <i>color picker</i>	45
Gambar 4.5 Hasil pengaturan dash-10-5-2-5.....	46
Gambar 4.6 Hasil tutorial membuat kotak	47
Gambar 4.7 Hasil fungsi lingkaran	48
Gambar 4.8 Hasil dari fungsi grafik	49

Gambar 4.9 Fungsi grafik dan pengaturannya	50
Gambar 4.10 Hasil tutorial menggambar kurva sinus.....	51
Gambar 4.11 Transformasi sistem koordinat.....	53
Gambar 4.12 Hasil tutorial 2c	56
Gambar 5.1 Hasil penambahan teks pada canvas.....	58
Gambar 5.2 Hasil fungsi teksHTML.....	60
Gambar 5.3 Input teks dan <i>popup</i>	62
Gambar 5.4 <i>Converter latex</i> menjadi <i>bitmap</i> PNG	66
Gambar 6.1 Hasil penambahan tombol pada canvas.....	68
Gambar 6.2 Hasil fungsi tombol dan <i>slider</i>	72
Gambar 6.3 <i>File</i> gambar bagian jangka sorong (<i>caliper</i>).....	74
Gambar 6.4 Hasil tutorial <i>drag and drop</i>	76
Gambar 6.5 Penjelasan tutorial 5.....	79
Gambar 7.1 Konsep animasi pada canvas.....	81
Gambar 7.2 Animasi roda gigi	83
Gambar 7.3 Parameter data untuk fungsi gear	84
Gambar 7.4 <i>Sprite</i> mesin	86
Gambar 7.5 <i>File</i> animasi_mesin.png	87
Gambar 7.6 Hasil tutorial animasi <i>sprite</i>	88
Gambar 8.1 Hasil tutorial penjumlahan vektor.....	93
Gambar 9.1 Hasil tutorial gerak dan kecepatan.....	101
Gambar 9.2 Hasil simulasi GLBB	103
Gambar 9.3 Model simulasi GLBB dengan beban massa	104
Gambar 9.4 Simulasi hubungan kecepatan dan massa.....	109
Gambar 9.5 Hasil simulasi gerak parabola.....	115
Gambar 9.6 Hasil simulasi gerak melingkar	122
Gambar 10.1 Hasil tutorial simulasi hukum Newton	130
Gambar 10.2 Simulasi keseimbangan beban pada mesin Atwood	131
Gambar 10.3 Hasil tutorial simulasi keseimbangan beban.....	136
Gambar 10.4 Struktur data untuk membentuk roda.....	137
Gambar 10.5 Simulasi katrol ganda.....	138
Gambar 10.6 Hasil tutorial simulasi katrol ganda	141
Gambar 11.1 Hasil tutorial konversi suhu	146
Gambar 11.2 Rancangan simulasi pengukuran panas.....	147
Gambar 11.3 Hasil tutorial pengukuran suhu.....	151

Gambar 12.1 Hasil tutorial lensa.....	159
Gambar 12.2 Hasil tutorial medan listrik.....	164
Gambar 13.1 Tampilan grafis judul aplikasi	166
Gambar 13.2 Grafis untuk latar belakang aplikasi (1200 x 600 <i>pixel</i>).....	166
Gambar 13.3 Contoh aset visual untuk tombol.....	167
Gambar 13.4 Agen pedagogis.....	167
Gambar 13.5 Gambar yang digunakan pada tutorial-20	171
Gambar 13.6 Hasil tutorial GUI fungsi halamanJudul	175
Gambar 13.7 Hasil tutorial GUI fungsi halamanNama.....	176
Gambar 13.8 Hasil tutorial GUI fungsi halamanMenu	176
Gambar 13.9 Hasil tutorial GUI fungsi halamanSimulasi	176
Gambar 13.10 Posisi tombol <i>home</i>	178
Gambar 13.11 Contoh Penambahan Instruksi Praktikum	180
Gambar 13.12 Sistem review praktikum dalam laboratorium virtual	180
Gambar 14.1 Mempublikasikan tutorial melalui <i>website</i>	181
Gambar 14.2 Penggabungan Lab-Virtual dengan LMS Moodle	182
Gambar 14.3 Tutorial-20 pada <i>Emulator</i> Android	182
Gambar 14.4 Pengembangan grafis 3 dimensi.....	183

BAB 1

Laboratorium Virtual

1.1 Konsep Dasar Laboratorium Virtual

Pembelajaran merupakan sebuah proses interaksi antara peserta didik dengan lingkungannya, sehingga terjadi perubahan tingkah laku ke arah yang lebih baik. Pembelajaran tersusun atas unsur manusiawi, material, fasilitas, perlengkapan dan prosedur yang saling mempengaruhi dalam mencapai tujuan tertentu [1]. Tujuan yang dimaksud adalah peserta didik memiliki kompetensi melalui upaya menumbuhkan serta mengembangkan sikap (*attitude*), pengetahuan (*knowledge*), keterampilan (*skill*) melalui sebuah proses pembelajaran yang tepat. Kata kunci dalam pencapaian tujuan ini adalah proses pembelajaran yang tepat.

Di bidang sains, terdapat salah satu aspek penting dalam proses pembelajaran yaitu pendekatan saintifik dimana peserta didik terlibat langsung atau mempunyai pengalaman terhadap benda-benda dan stimulus-stimulus dalam lingkungan belajar^{[2][3]}. Pendekatan saintifik (*scientific approach*) adalah model pembelajaran yang menggunakan kaidah-kaidah keilmuan yang memuat serangkaian aktivitas pengumpulan data melalui proses mengamati (*observing*), menanya (*questioning*), mencoba (*experimenting*), mengolah data atau informasi dilanjutkan dengan menganalisis, menalar (*associating*), dan menyimpulkan, menyajikan data atau informasi (mengomunikasikan/*communicating*), dan menciptakan serta membentuk jaringan (*networking*)^[4].

Pendekatan saintifik pada hakekatnya melibatkan dimensi produk dan proses. Dimensi produk merupakan kumpulan teori yang telah teruji kebenarannya, sedangkan dimensi prosesnya, merupakan langkah-langkah yang harus ditempuh untuk memperoleh pengetahuan atau gejala-gejala alam yang sering dikenal sebagai metode ilmiah yang berlandaskan pada sikap ilmiah. Terkait dengan hal tersebut pembelajaran sains perlu berorientasi pada keterampilan proses. Peter C. Gega ^[5] merumuskan enam keterampilan proses yang dikembangkan dalam pembelajaran sains yang dalam hal ini dikhususkan pada bidang fisika, yaitu: mengamati,

mengklasifikasi, mengukur, mengkomunikasikan, menyimpulkan, dan merencanakan eksperimen.

Keterkaitan pengamatan eksperimental dengan teori adalah dua hal yang mendasari pemahaman yang koheren di bidang sains. Pemahaman teoritis konsep-konsep sains perlu dilengkapi dengan pengalaman praktis di laboratorium. Laboratorium merupakan tempat bagi peserta didik untuk melakukan eksperimen-eksperimen dari teori yang telah diberikan di kelas. Fungsi dari eksperimen itu sendiri sebagai penunjang pembelajaran guna meningkatkan pemahaman peserta didik terhadap suatu materi yang telah dipelajari. Laboratorium selama ini telah menjadi jantung dari pendidikan sains. Laboratorium setidaknya memiliki tiga fungsi dasar, yaitu sebagai (1) sumber belajar, dimana laboratorium digunakan untuk memecahkan masalah yang berkaitan dengan ranah kognitif, afektif dan psikomotorik dengan melakukan percobaan, (2) metode pendidikan, yang meliputi metode pengamatan dan metode percobaan, dan (3) sarana penelitian, yaitu tempat dilakukannya berbagai penelitian sehingga terbentuk pribadi peserta didik yang bersikap ilmiah.

Di sisi lain, perkembangan teknologi pada 2 dekade terakhir membawa berbagai macam perubahan di segala bidang, termasuk di bidang pendidikan. Teknologi digital mempermudah proses pembelajaran dengan menghadirkan format baru dalam mendukung proses pembelajaran. Era digital yang mendorong inovasi-inovasi baru dimana teknologi telah membuka kemungkinan proses pembelajaran sains baru dalam bentuk laboratorium virtual. Laboratorium virtual merupakan sebuah pengalaman belajar yang mensimulasikan laboratorium otentik. Laboratorium disimulasikan dan divisualisasikan melalui format digital, maka dapat digunakan siswa untuk mengeksplorasi konsep dan teori.

Laboratorium virtual dapat didefinisikan sebagai perangkat lunak multisensori yang memiliki interaktivitas untuk mensimulasikan praktikum-praktikum tertentu dengan mereplikasi laboratorium konvensional. Laboratorium virtual memungkinkan siswa untuk belajar melalui pendekatan studi kasus, berinteraksi dengan peralatan laboratorium, melakukan eksperimen, menganalisis eksperimen sekaligus mengevaluasi proses yang dilakukan. Siswa dapat melihat ke dalam perangkat yang mereka operasikan melalui tampilan visual, animasi dan representasi yang diadaptasi dari laboratorium yang sesungguhnya. Sehingga dapat

dikatakan bahwa dengan laboratorium virtual, kemungkinan untuk menjelajahi, berekspresi, dan belajar menjadi lebih dinamis.

Dalam penggunaannya, laboratorium virtual membutuhkan perangkat keras yang mendukung input tertentu dari penggunanya. Input yang dimaksud antara lain: menekan tombol, menyentuh layar, atau melalui gerakan anggota tubuh pengguna (*gesture*). Jenis perangkat keras untuk mengoperasikan laboratorium virtual semakin beragam seiring perkembangan teknologi, mulai dari komputer, konsol (*Digital Player Console*), proyeksi dinding CAVE (*Cave Automatic Virtual Environment*), gawai (*mobile phone*), dan perangkat realitas virtual (*Head Mounted Display*).

Perangkat untuk menjalankan laboratorium virtual tersebut dapat menciptakan pengalaman 'imersi rendah' (*low immersion*), maupun menghasilkan pengalaman 'imersi tinggi' (*high immersion*). Pada simulasi laboratorium yang dijalankan pada komputer/*desktop* pada umumnya menghasilkan lingkungan pembelajaran interaktif dengan imersi rendah. Sedangkan penggunaan perangkat realitas virtual (*Virtual Reality*) akan menghasilkan imersi tinggi, yaitu sebuah pengalaman mendalam yang menempatkan pengguna di dalam sebuah lingkungan maya yang sepenuhnya mengelilingi pengguna (360°) sehingga pengguna merasa bahwa dirinya berada di dalamnya dan bagian dari itu.



Gambar 1.1 Laboratorium Virtual PhET (*low immersion*)

(sumber : <https://phet.colorado.edu/>)

Pada laboratorium virtual fisika PhET di atas, simulasi dijalankan melalui platform komputer desktop dalam tampilan grafis 2 dimensi. Pengguna dapat menjalankan praktikum melalui layar komputer dengan input interaktivitas yang berasal dari tombol *keyboard* atau *mouse*. Pada laboratorium virtual jenis ini, pengalaman yang

didapatkan pengguna disebut sebagai pengalaman “imersi rendah”, karena beberapa kegiatan praktikum diwakili dengan menekan tombol, menyentuh atau menggeser *mouse*. Interaksi antara peralatan laboratorium dan pengguna dibatasi oleh ruang kerja yang datar, dan input yang relatif berbeda dengan kondisi riil. *Keyboard*, mouse atau sentuhan jari menjadi perpanjangan tangan untuk melakukan kegiatan praktikum, sehingga bisa dikatakan pengguna tidak “menyentuh” secara langsung peralatan laboratorium.



Gambar 1.2 Simulasi International Space Station ISS NASA

(Sumber : <https://www.oculus.com/experiences/quest/>)

Pada simulasi *International Space Station* (ISS) NASA di atas, simulasi dijalankan melalui perangkat HMD (*Head Mounted Display*) yang mampu menghasilkan sebuah lingkungan maya yang mereplika kondisi ISS yang sesungguhnya. Pengguna akan merasakan pengalaman berada di dalam ISS dan dapat berinteraksi dengan beberapa perangkat yang ada di dalamnya. Pengguna dapat menggunakan tangan untuk memegang benda, dimana “tangan virtual” akan muncul dan merepresentasikan tangan pengguna yang sesungguhnya. Gerakan kepala dan kaki pengguna juga dapat menghasilkan respon pada lingkungan virtual, sehingga pengalaman (imersi) yang didapatkan pengguna mendekati nyata (*high immersion*).

Pada aplikasi laboratorium virtual yang menggunakan perangkat HMD, seperti Labster, pengguna dapat “memasuki” laboratorium secara virtual dan mengakses beberapa jenis praktikum, lengkap dengan peralatan dan prosedur kerja. Praktikum dikemas dengan pendekatan studi kasus, memiliki sebuah alur cerita yang dapat diikuti oleh pengguna, dan dapat mengevaluasi hasil praktikum yang dilakukan pengguna. Dengan merujuk beberapa contoh laboratorium virtual tersebut, maka

kita dapat mendefinisikan beberapa tujuan dan keunggulan penggunaan laboratorium virtual.



Gambar 1.3 Laboratorium Virtual Labster

(sumber: <https://labster.com>)

1.2 Tujuan dan Keunggulan Penggunaan Laboratorium Virtual

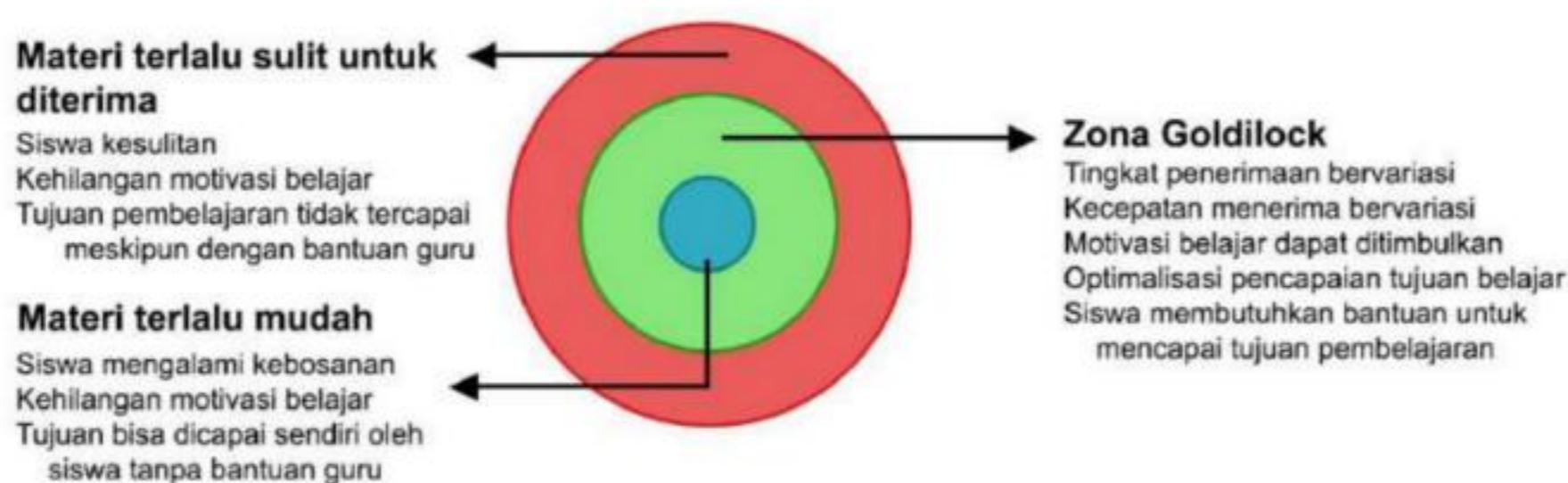
Popularitas laboratorium virtual dalam 1 atau 2 dekade terakhir, tidak terlepas dari perkembangan teknologi dan kemampuannya dalam mereplikasi kondisi laboratorium yang sesungguhnya dengan penambahan elemen-elemen baru yang lebih menarik untuk diikuti oleh siswa. Secara spesifik, beberapa tujuan dan keunggulan dari penggunaan laboratorium virtual adalah sebagai berikut :

1. Personalisasi Pembelajaran

Laboratorium virtual menyediakan sarana teknologi untuk menghadirkan pendekatan pembelajaran yang lebih personal dan khusus, dengan mempertimbangkan kecepatan dan kebutuhan pembelajaran individu. Dalam teori Perkembangan Proksimal oleh Vygotsky (1978), seorang siswa memiliki sebuah zona optimal ("zona Goldilocks") dalam menerima proses pembelajaran. Seorang pengajar seharusnya tidak meminta siswa untuk melakukan hal-hal yang mereka rasa tidak mungkin untuk dilakukan (terlalu sulit) dan pengajar juga tidak perlu meminta siswa untuk melakukan hal-hal yang sudah dapat mereka lakukan dan temukan dengan sangat mudah (terlalu mudah). Pada zona yang tepat siswa dapat diajak untuk mempelajari hal-hal baru dengan cepat.

Namun yang perlu diperhatikan adalah, setiap siswa memiliki zona yang berbeda. Gaya belajar mereka sendiri yang unik, dan pembelajaran mandiri sangat membantu untuk mempersonalisasikan pendidikan bagi pelajar yang beragam. Disinilah diperlukan personalisasi pembelajaran, siswa perlu diberikan kesempatan yang bervariasi dalam mempelajari sesuatu.

Dalam laboratorium konvensional, proses praktikum pada umumnya akan dilakukan secara seragam dengan metode yang ditentukan oleh pengajar. Kelemahan metode ini cenderung mengasingkan siswa yang berkinerja lebih rendah, padahal sebenarnya menurut teori perkembangan proksimal, mereka hanya memiliki gaya belajar yang berbeda dengan siswa yang lain dan membutuhkan metode yang berbeda. Laboratorium virtual dapat mendukung siswa dengan gaya belajar yang berbeda dengan memberikan pengalaman multi-indera. Siswa dapat memahami suatu praktikum dengan lebih bebas, memahami metode praktikum dengan lebih leluasa, tidak perlu khawatir dengan kesalahan, dan dapat mengulang berkali-kali sebuah praktikum sampai mampu menyimpulkan sendiri metode yang tepat. Dengan kata lain, laboratorium virtual dapat menjadi alat untuk pra-praktikum, sebelum siswa benar-benar menjalankan praktikum yang sesungguhnya di laboratorium konvensional.



Gambar 1.4 Zona Goldilock

2. Mengatasi sumberdaya yang terbatas

Laboratorium virtual dapat menyediakan laboratorium lengkap dengan peralatan yang mahal, namun dengan biaya rendah. Hal ini dapat menjadi solusi yang menarik bagi sekolah yang memiliki sumber daya terbatas, ruang, dan sarana prasarana fasilitas laboratorium terbaru. Melalui laboratorium virtual, siswa dapat memiliki akses tanpa batas ke pengaturan laboratorium yang canggih dan peralatan mahal

seperti *fermentor*, *sequencer*, mikroskop, dan perangkat generasi baru yang biasanya mereka tidak memiliki kesempatan untuk menggunakannya.

Dalam praktikum tertentu, peralatan yang digunakan terkadang sangat mahal, misalnya mikroskop confocal. Biaya satu mikroskop confocal berkisar antara 100 juta sampai 2 Milyar, hal ini tentu saja tidak terjangkau bagi sebagian besar sekolah. Laboratorium virtual dapat mensimulasikannya dengan tepat, dan dapat diduplikasi dengan mudah, sehingga tidak diperlukan biaya yang tinggi. Dalam kasus lain, sebuah sekolah mungkin memiliki sebuah perangkat praktikum namun dalam jumlah yang terbatas, dan siswa harus menunggu giliran untuk dapat mengoperasikannya. Cela inilah yang dapat ditutupi oleh laboratorium virtual dimana aplikasi dapat dijalankan selama waktu tunggu eksperimen, selama menunggu siswa dapat menjalankan perangkat tersebut dalam versi virtual, memahami teknis dan cara operasionalnya, dan ketika gilirannya tiba untuk menggunakan perangkat yang sesungguhnya, siswa akan jauh lebih mudah memahami perangkat tersebut.



Gambar 1.5 Mikroskop confocal virtual

(sumber : www.labster.com)

3. Visualisasi yang Kompleks

Dengan laboratorium virtual, siswa dapat melihat apa yang tidak terlihat. Siswa tidak terbatas pada deskripsi kata-kata atau ilustrasi pada buku. Kemampuan untuk memvisualisasikan fungsi atau mekanisme kompleks yang tidak terlihat oleh mata telanjang akan membuat sebuah konsep lebih mudah untuk dipahami. Sebagai contoh, animasi 3D tentang mesin kendaraan bermotor memungkinkan siswa untuk

melihat ke dalam mesin tentang proses terjadinya perubahan energi di dalam mesin. Siswa dapat melihat lebih detil bagian-bagian mesin dan bagaimana peranannya dalam sebuah sistem kerja. Dengan laboratorium virtual, siswa juga dapat merekayasa waktu, melihat suatu proses lebih lambat atau lebih cepat.

Visualisasi grafis dalam laboratorium virtual memungkinkan representasi abstrak berubah menjadi pengalaman yang lebih konkret yang mengaktifkan pikiran siswa dalam memahami sebuah konsep tertentu, dan memungkinkan pembelajaran melalui pengalaman. Selain itu, ini dapat memungkinkan pelajar untuk berkonsentrasi pada konsep sentral tanpa gangguan.



Gambar 1.6 Visualisasi sel dalam format 3 dimensi pada lab virtual

4. Pembelajaran berbasis kasus

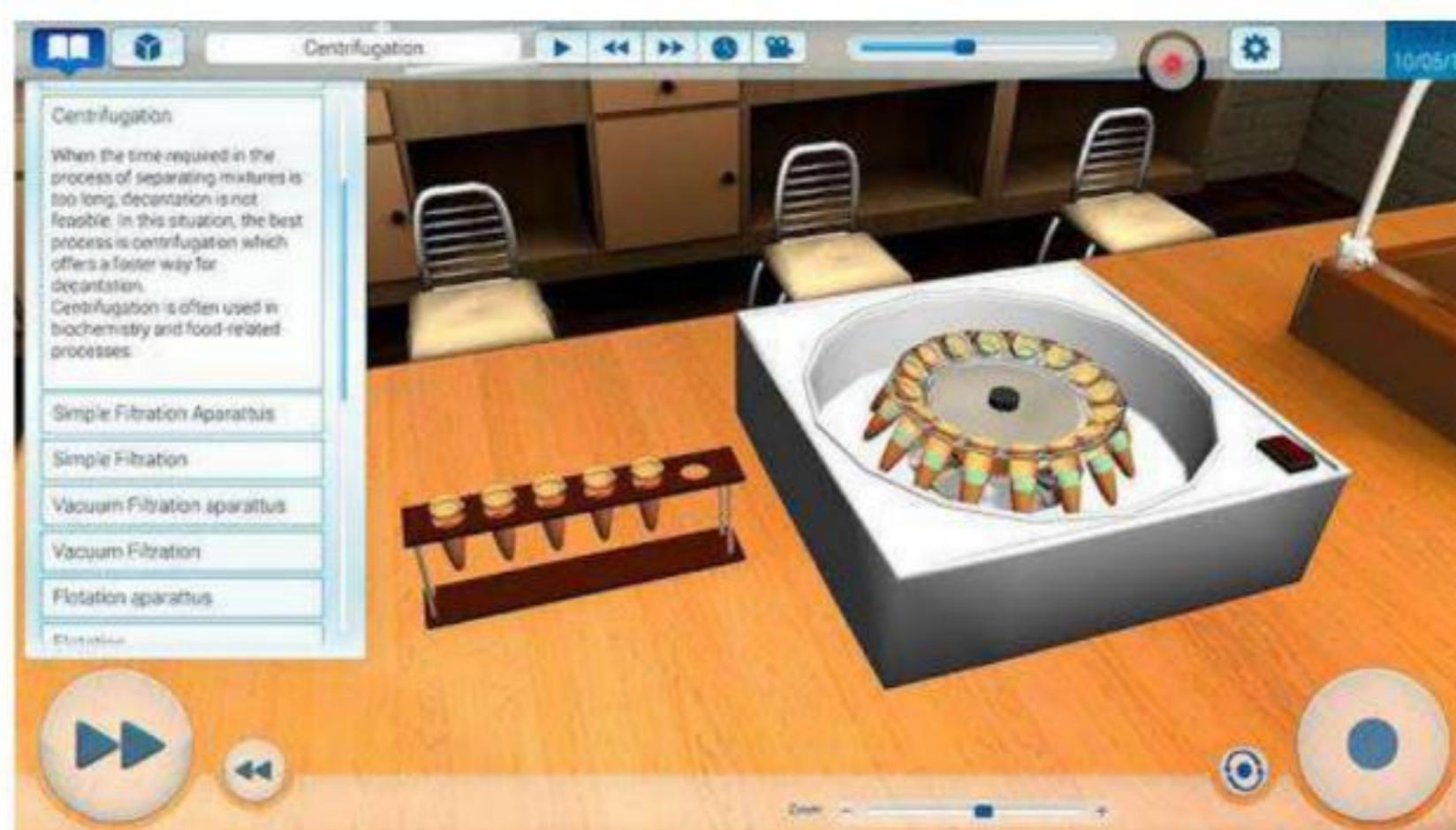
Aplikasi laboratorium virtual, pada umumnya memiliki sebuah tema yang spesifik. Memiliki pilihan eksperimen, dan tidak jarang memiliki praktikum berdasarkan kasus tertentu. Kemampuan aplikasi untuk menghitung berbagai macam kondisi dan memberikan umpan balik atas kondisi tersebut, menjadikan aplikasi laboratorium virtual dapat menjadi alternatif untuk mempelajari sesuatu secara mendetail. Sebagai contoh, dalam simulasi energi listrik dengan objek magnet dan kumparan tembaga, seorang siswa dapat mencoba berbagai macam posisi, arah dan gerakan magnet untuk mengetahui medan gaya yang ditimbulkan. Siswa juga dapat mencoba merubah berbagai variabel yang terlibat dalam praktikum tersebut, sehingga pemahaman yang dihasilkan terhadap praktikum yang dilakukan dimungkinkan lebih baik dibandingkan dengan praktikum yang menggunakan alat yang tidak bervariasi.

5. Fleksibilitas Waktu

Dalam laboratorium virtual variabel waktu dapat diatur sedemikian rupa, dipercepat, diperlambat, dihentikan, atau diulang. Fleksibilitas waktu ini penting untuk proses pembelajaran yang efektif. Siswa dapat mengulangi praktikum di

waktu luang mereka untuk membantu mereka menguasai konsep-konsep sulit atau menyegarkan ingatan mereka. Untuk praktikum yang membutuhkan waktu yang lama, misalkan pertumbuhan sel, simulasi dapat dipercepat sehingga siswa dapat memahami tahapan-tahapan yang terjadi tanpa harus menunggu sesuai dengan waktu yang sebenarnya.

Ketika diterapkan di sebuah kelas, laboratorium virtual dapat juga menjadi media pra-praktikum, dimana siswa telah mengetahui prosedur praktikum sebelumnya melalui simulasi, sehingga ketika menghadapi praktikum yang sebenarnya waktu akan menjadi lebih optimal, karena siswa telah mengetahui langkah-langkah yang tepat untuk dilakukan.



Gambar 1.7 Fitur kontrol waktu pada laboratorium virtual
(sumber : aplikasi V-Lab Evobook)

6. Menjadikan kegagalan sebagai hal yang produktif

Di laboratorium virtual, siswa dapat menjalankan praktikum dengan berbagai kondisi, yang memungkinkan terjadinya kesalahan praktikum. Simulasi akan menganalisis kondisi dan kesalahan yang dilakukan, kemudian memberikan umpan balik kepada siswa. Pada umumnya ketika terjadi kegagalan, aplikasi akan memberikan penilaian dan akan mempersilahkan siswa untuk mengulang praktikum tersebut. Dengan kata lain, setelah mengetahui kegagalan pada praktikum pertama, siswa dapat mencoba kondisi baru di praktikum kedua dan seterusnya. Bahkan aplikasi laboratorium virtual yang canggih dapat memberikan umpan balik yang realistik terhadap sebuah kesalahan/kegagalan. Sebagai contoh jika siswa membuat

kesalahan dalam simulasi kimia, mereka dapat menyebabkan ledakan dan terkena asam di mata mereka, maka simulasi akan menampilkan efek penglihatan buram di layar untuk beberapa waktu. Dengan cara ini, siswa akan menjadi lebih waspada terhadap langkah-langkah yang diambil pada praktikum berikutnya, terlebih lagi ketika praktikum pada laboratorium yang sesungguhnya.



Gambar 1.8 Kesalahan praktikum dalam laboratorium virtual

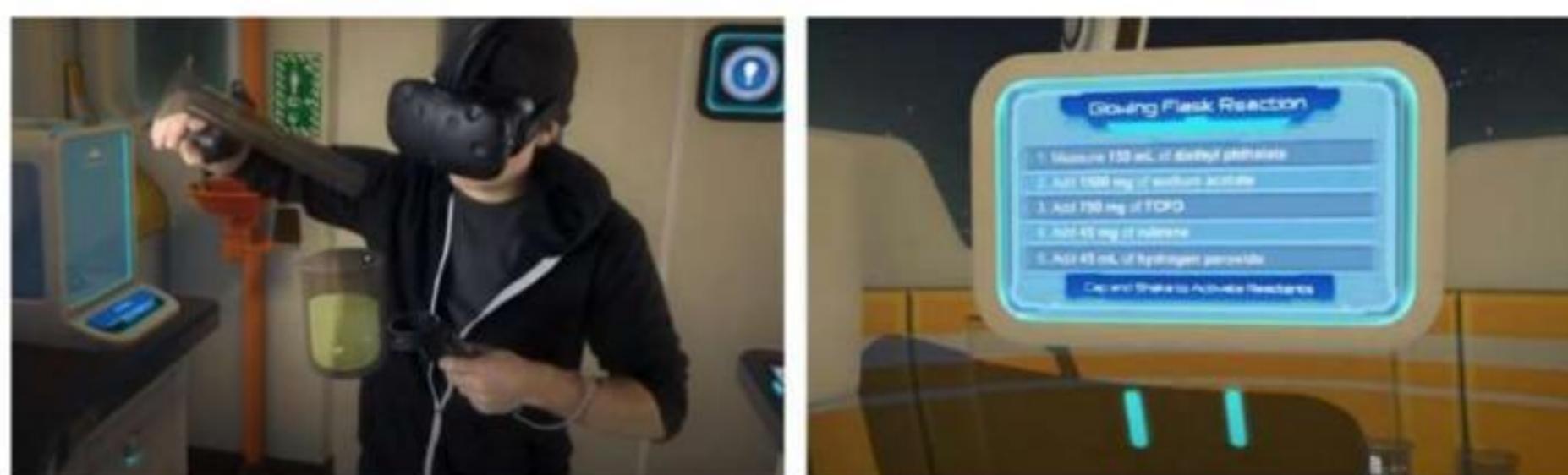
(Sumber : BYU Virtual Lab)

7. Praktikum menjadi menyenangkan melalui gamifikasi

Pada era ini, minat belajar siswa sangat fluktuatif dan lebih sering minat tersebut berada pada posisi rendah. Berbeda dengan minat dalam bermain, yang mana lebih cenderung tinggi. Bermain (*bermain gadget*) pada era ini menjadi sangat menyenangkan bagi siswa karena adanya gamifikasi. Permainan menjadi sistem dimana pemain berpartisipasi dalam tantangan abstrak, ditentukan oleh aturan, interaktivitas dan umpan balik, yang menghasilkan suatu hasil diukur dan sering menimbulkan reaksi emosional. Hal inilah yang mendorong motivasi tinggi dalam bermain.

Berbeda dengan laboratorium konvensional, dalam laboratorium virtual faktor gamifikasi bisa ditambahkan. Seorang siswa dapat memiliki level tertentu dan nilai tertentu, yang akan naik seiring kesuksesan dalam praktikum, dan akan membuka praktikum-praktikum baru dengan tingkat kesulitan yang lebih tinggi. Laboratorium virtual juga dapat menampilkan agen pedagogik, yang memberikan misi atau cerita tertentu yang dapat diikuti oleh siswa. Pada tahapan selanjutnya, secara umum ketika siswa menikmati melakukan sesuatu, dan menemukan subjek yang mereka

sukai, mereka akan termotivasi secara intrinsik, berinvestasi lebih banyak di dalamnya, dan melakukannya dengan lebih baik.



Gambar 1.9 Gamifikasi pada SuperChem VR

(Sumber : Schell Games)

Dalam literatur lain, akan ditemukan kelebihan dan kelemahan penggunaan laboratorium virtual dalam pembelajaran sains. Pertanyaan terkait efektivitas laboratorium virtual dalam pendidikan, dan bagaimana cara mengukur efektivitasnya dapat menjadi diskusi panjang yang tidak dapat diulas secara mendalam pada buku ini. Buku ini mencoba untuk melihat dari sisi positif, yaitu nilai tambah apa yang dapat kita bawa dari keberadaan laboratorium virtual untuk membantu siswa dalam mencapai tujuan pembelajaran.

1.3 Terminologi dalam Laboratorium Virtual

Dalam penggunaan istilah laboratorium virtual, maka kita akan mengaitkannya dengan beberapa istilah lain yang memiliki kedekatan baik itu dari segi definisi atau dari segi operasional. Setidaknya dalam buku ini terdapat beberapa istilah yang terkait, yaitu :

- **Simulasi**

Simulasi merupakan suatu proses peniruan dari sesuatu yang nyata beserta lingkungan di sekelilingnya. Peniruan yang dimaksud adalah menggambarkan sifat-sifat karakteristik kunci dari objek yang ditiru. Simulasi dapat digunakan untuk menunjukkan efek yang terjadi akibat beberapa kondisi atau alternatif tindakan terhadap suatu sistem. Simulasi digunakan ketika sistem yang rill tidak dapat dijalankan, sulit/tidak dapat diakses, dalam lingkungan yang berbahaya atau keterlibatan manusia di lingkungan tersebut beresiko tinggi, sistem yang sedang dirancang dalam proses pengembangan atau bahkan sistem yang belum dibuat. Dalam ruang lingkup laboratorium virtual, simulasi merupakan peniruan dari perangkat atau

kegiatan praktikum, simulasi juga memungkinkan alternatif-alternatif akibat yang terjadi apabila pengguna melakukan kegiatan di luar prosedur yang ditentukan.

- **Imersi**

(*immersion*, terjemahan: perendaman) merupakan persepsi pengguna yang merasakan kehadiran secara fisik di dunia non-fisik (maya). Persepsi dibuat dengan menyajikan pengalaman indrawi melalui gambar, suara atau rangsangan lain. Dalam ranah realitas maya (*virtual reality*) misalnya, imersi dibuat dengan menempatkan pengguna di lingkungan maya secara menyeluruh, melalui perangkat HMD (*Head Mounted Display*) pengguna dapat melihat ke segala arah dan secara psikologis akan merasa berada di dalam lingkungan maya tersebut.

- **Augmented Reality**

Augmented reality (AR) adalah pengalaman visual interaktif di mana objek yang berada di dunia nyata dan ditambahkan informasi baru oleh sebuah perangkat untuk menghasilkan persepsi baru. Visualisasi lingkungan fisik dari dunia nyata dikombinasikan dengan visualisasi maya untuk menghadirkan persepsi baru. Dalam definisi ini *augmented reality* harus memenuhi tiga fitur dasar, yaitu: kombinasi dunia nyata dan virtual, interaksi terjadi secara langsung, dan visualisasi yang akurat dalam penggabungan antara objek virtual dan objek nyata.



Gambar 1.10 AR Tata Surya menggunakan *marker* (penanda)

(Sumber : EON Solar System AR)

- **Virtual Reality**

Virtual reality (VR, terjemahan : realitas maya) adalah sebuah teknologi yang menempatkan pengguna di lingkungan maya. Realitas maya ditampilkan melalui peralatan khusus yang memungkinkan pengguna untuk melihat

dunia buatan/maya, bergerak di dalamnya, dan berinteraksi dengan fitur atau barang virtual. Perangkat khusus yang dimaksud antara lain: kombinasi beberapa layar datar (minimal 4 arah layar) dan perangkat *Head Mounted Display* (HMD).

Hal utama yang membedakan antara VR dan AR, adalah di dalam VR seluruh lingkungan dibentuk oleh perangkat, keberadaan objek fisik dari dunia nyata sudah tidak ada lagi, berbeda dengan AR yang menggabungkan objek nyata dan objek maya. Dalam ranah laboratorium virtual, VR dipakai oleh beberapa pengembang aplikasi untuk menampilkan laboratorium virtual secara menyeluruh seperti Labster, CTIL, Digihuman, LAVREB dan beberapa laboratorium virtual yang dikembangkan oleh universitas-universitas terkemuka.

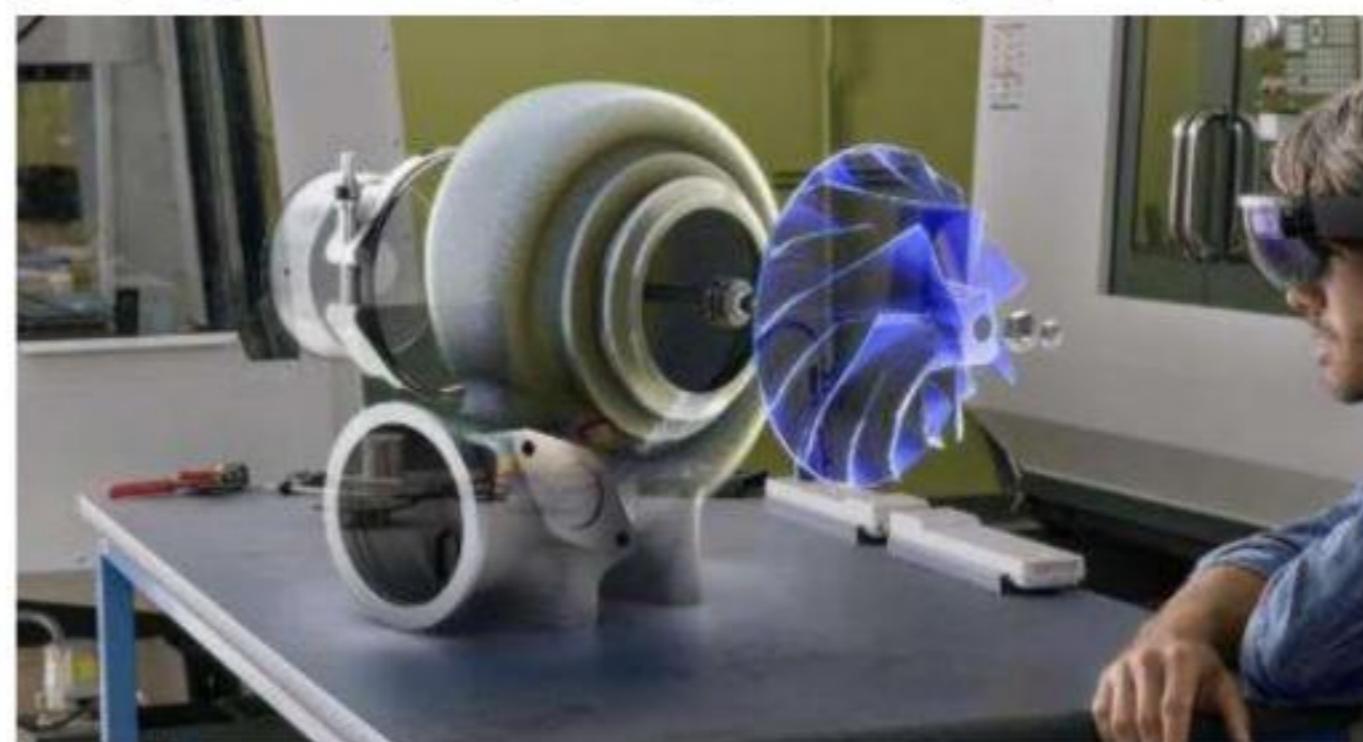


Gambar 1.11 Penggunaan HMD
untuk menjalankan aplikasi laboratorium virtual
(Sumber : utoronto.ca)

- ***Mixed Reality***

Mixed reality (MR, terjemahan: realitas campuran) adalah penggabungan dunia nyata dan virtual untuk menghasilkan lingkungan dan visualisasi baru, di mana objek fisik dan digital ditampilkan bersama-sama secara berdampingan dan berinteraksi. Realitas campuran tidak secara eksklusif terjadi di dunia fisik atau virtual, tetapi merupakan gabungan antara realitas nyata dan realitas maya. Sehingga, bisa dikatakan bahwa MR merupakan penggabungan antara teknologi VR dan AR. Dalam MR pengguna dapat bergerak secara langsung di lingkungan fisik, bersamaan dengan hal tersebut pengguna dapat melihat objek-objek maya yang ditambahkan oleh perangkat, sekaligus berinteraksi dengannya. Perangkat MR mampu menangkap lingkungan fisik secara baik, dan menggabungkan objek virtual secara presisi ke dalamnya. Melalui HMD khusus, pengguna akan melihat

penggabungan antara objek virtual dan objek nyata dan menghasilkan persepsi baru yang lebih dinamis. Pada saat buku ini ditulis, terdapat beberapa perangkat MR seperti Hololens, Acer Windows *Mixed Reality*, Samsung Odyssey dan beberapa perangkat dalam pengembangan.



Gambar 1.12 Penggunaan *Mixed Reality*

(sumber : <https://wonder.store/microsoft/microsoft-hololens/>)

1.4 Mendesain dan Mengembangkan Laboratorium Virtual

A. Metodologi pengembangan laboratorium virtual

Pengembangan laboratorium virtual, pada dasarnya termasuk ke dalam kategori penelitian pengembangan berbasis produk atau lebih dikenal dengan istilah *Research and Development* (R&D). Mengacu pada konsep R&D menurut Borg dan Gall^[6], penelitian riset dan pengembangan adalah metode penelitian yang digunakan untuk menghasilkan produk tertentu, dan menguji keefektifan produk tersebut. Terkait dengan pengembangan laboratorium virtual sebagai media pembelajaran, model pengembangan yang secara umum dipakai adalah model prosedural Dick and Carrey yang mana menekankan pada penerapan prinsip desain pengembangan yang disesuaikan dengan langkah-langkah berurutan. Model prosedural Dick and Carey ini merupakan model penelitian yang berorientasi pada pemaparan tahapan penelitian secara deskriptif yang terdiri atas tiga tahapan yakni tahap pra-pengembangan, pengembangan, dan pasca-pengembangan.

Penelitian pengembangan digunakan untuk menghasilkan produk tertentu, dan untuk menyempurnakan suatu produk yang sesuai dengan acuan dan kriteria dari produk yang dibuat, sehingga menghasilkan produk yang baru melalui berbagai tahapan dan validasi atau pengujian. Merujuk Borg dan Gall serta Dick dan Carrey,

metode pengembangan laboratorium virtual dapat divisualisasikan pada bagan berikut :



Gambar 1.13 Metode pengembangan laboratorium virtual

B. Pengetahuan dasar dalam pengembangan laboratorium virtual

Laboratorium virtual merupakan produk multi disiplin ilmu yang membutuhkan kolaborasi tim pengembang. Pemahaman terhadap kondisi laboratorium yang sebenarnya, operasional dan teknik perangkat praktikum, pemahaman akan simulasi sains yang akan ditampilkan, kemampuan grafis untuk memvisualisasikan objek, dan kemampuan pemrograman untuk membangun aplikasi, menjadikan pengembangan laboratorium visual sangat kompleks dan perlu perencanaan tim yang baik. Secara ringkas, pengetahuan dasar yang diperlukan dalam pengembangan laboratorium virtual adalah :

1. Pengetahuan dasar tentang prosedur praktikum, perangkat laboratorium dan operasionalnya. Tim pengembang membutuhkan tenaga ahli di bidang pendidikan sains yang dapat mendeskripsikan secara detail tentang proses praktikum dan berbagai tindakan yang dapat dilakukan oleh pengguna dalam praktikum, serta akibat dari tindakan yang dilakukan terhadap jalannya praktikum.
2. Pengetahuan untuk mengadaptasi praktikum dalam bentuk simulasi. Berdasarkan pengalaman penulis, sering kali di lapangan sulit untuk

menemukan titik temu antara pengajar sains (guru atau dosen) dengan pengembang aplikasi. Hal ini secara umum diakibatkan karena ketidaksepahaman dalam mengadaptasi praktikum atau materi sains dalam laboratorium nyata ke dalam bentuk simulasi. Tim pengembang membutuhkan tenaga ahli di bidang interaktivitas, yang memahami keunggulan dan keterbatasan tim dalam mewujudkan simulasi. Tenaga ahli tersebut harus memahami apa yang dapat dilakukan dan apa yang tidak, serta berkomunikasi dengan baik dengan menjembatani antara pembuat materi praktikum (pengajar sains) dan tim pengembang simulasi (programer)

3. Pengetahuan dasar dalam mengembangkan grafis. Laboratorium virtual membutuhkan visualisasi yang menarik, karena elemen visual merupakan salah satu nyawa utama dari aplikasi virtual. Tim pengembang membutuhkan tenaga ahli di bidang grafis 2 Dimensi maupun 3 Dimensi. Kemampuan ini harus didukung dengan penguasaan *software* grafis 2D seperti Photoshop, ilustrator, gimp dan sebagainya atau *software* grafis 3D seperti Blender, 3Ds Max dan sebagainya.
4. Pengetahuan dasar pemrograman. Untuk mewujudkan aplikasi laboratorium virtual dibutuhkan penguasaan teknis dalam pemrograman. Pengembangan laboratorium virtual secara teknis dapat dilakukan dengan menggunakan berbagai aplikasi tergantung tingkat teknologi yang akan dihadirkan dalam laboratorium virtual tersebut. Saat buku ini ditulis, setidaknya terdapat beberapa perangkat lunak populer yang dapat digunakan untuk mengembangkan aplikasi laboratorium virtual, di antaranya :
 - a. Unity3D
 - b. Unreal Engine
 - c. Android Studio
 - d. Flash/Animate
 - e. Smart Apps Creator
 - dan sebagainya

Kemampuan mengoperasikan aplikasi di atas tentusaja membutuhkan penguasaan terhadap bahasa pemrograman seperti C#, Javascript, HTML, Actionscript atau sejenisnya. Sehingga dibutuhkan tenaga ahli di bidang pemrograman.

1.5 Pengembangan Laboratorium Virtual dalam Buku ini

Tingkatan pengembangan laboratorium virtual sangat beragam dari sebuah sistem simulasi yang sederhana, sampai simulasi kompleks yang melibatkan lingkungan maya dengan berbagai perangkat di dalamnya. Kompleksitas tersebut tentusaja tidak dapat untuk dijabarkan dalam buku ini. Oleh karena itu buku ini secara spesifik akan membahas pengembangan laboratorium virtual dalam bentuk simulasi-simulasi sederhana dengan topik praktikum fisika.

Pengembangan aplikasi laboratorium virtual akan menggunakan aplikasi berbasis HTML 5 canvas dengan bahasa pemrograman Javascript. Buku akan membahas beberapa simulasi fisika dasar dengan memanfaatkan *library* yang sudah dikembangkan penulis, sehingga proses pemrograman relatif menjadi lebih sederhana dengan jumlah baris kode yang tidak terlalu banyak (50 – 200 baris).

Buku ini berusaha menjelaskan teknis sederhana untuk mengadaptasi sebuah praktikum fisika menjadi bentuk simulasi digital. Rumus-rumus fisika terkait praktikum akan diadaptasi ke dalam bentuk kode, dengan harapan ketika pembaca memahami proses adaptasi tersebut, maka teknik yang sama dapat diterapkan pada pengembangan laboratorium virtual yang lebih kompleks.

Simulasi yang dibuat dalam buku ini termasuk kedalam kategori simulasi 2D dengan imersi rendah. Produk simulasi pada akhirnya dapat dijalankan melalui internet browser atau dipublish menjadi aplikasi Android. Simulasi juga dapat dikombinasikan dengan LMS (*Learning Management System*) berbasis HTML.

Dari segi grafis, buku tidak akan membahas secara detail tentang pengembangan grafis tingkat tinggi, melainkan akan membahas teknik membangun grafis sederhana dengan kode grafis yang telah tersedia pada *library*. Pada bab akhir juga akan dijelaskan penambahan *Graphic Unit Interface* untuk menjadikan aplikasi laboratorium virtual menjadi lebih baik dari segi tampilan visual (*User Interface*) maupun pengalaman penggunaan (*User Experience*).



File sumber dan tutorial yang digunakan di dalam buku ini dapat diakses pada link berikut : <https://www.wandah.org/laboratorium-virtual>

1.6 Referensi Pengembangan Laboratorium Virtual

Dalam proses penyusunan buku dan material pengembangan laboratorium virtual di dalam buku ini, digunakan beberapa referensi utama sebagai berikut :

1. PhET Simulation

PhET Simulation merupakan kumpulan simulasi praktikum virtual yang dikembangkan oleh University of Colorado. Beberapa simulasi praktikum fisika yang diadaptasi dalam buku ini diperoleh dari link berikut :

<https://phet.colorado.edu/en/simulations/category/physics>

2. Simulasi Fisika oleh Andrew Duffy - Boston University

Halaman yang dikembangkan oleh Andrew Duffy memiliki lebih dari 200 jenis simulasi fisika. Beberapa simulasi praktikum fisika di buku ini diadaptasi dari simulasi Andrew Duffy dan disederhanakan dengan fungsi berbahasa Indonesia. Halaman simulasi Andrew Duffy dapat dibuka pada link berikut :

<http://physics.bu.edu/~duffy/classroom.html>

3. Labster

Labster merupakan aplikasi laboratorium virtual komersial dengan kualitas tinggi, menghadirkan berbagai praktikum dan telah digunakan di beberapa Universitas maupun Sekolah Menengah di seluruh dunia. Beberapa referensi terkait penggunaan laboratorium virtual, kelebihan dan prosedur penggunaannya diadaptasi dalam buku ini dengan beberapa penyesuaian.

1.7 Lisensi Buku

Buku ini diterbitkan secara *online* dalam bentuk *e-book* dan diedarkan secara gratis, dengan beberapa ketentuan sebagai berikut :

1. Pembaca diperbolehkan menyebarkan dan memperbanyak buku tanpa mengubah, menambah atau mengurangi konten buku.
2. Pembaca diperbolehkan mencetak buku untuk keperluan pribadi, namun tidak diperbolehkan mencetak untuk diperjual belikan (dilarang mengomersialkan buku ini dalam bentuk apapun).
3. Penambahan atribut nama penulis (*credit*) diperlukan pada setiap aplikasi yang dihasilkan.
4. Pembaca diperbolehkan menggunakan dan memodifikasi *file* tutorial untuk keperluan apapun, termasuk keperluan komersial.

BAB 2

Teknologi HTML 5 dan Canvas

2.1 HTML Canvas

Dalam membuat sebuah simulasi, kita dapat memanfaatkan beberapa *software* atau bahasa pemrograman yang mendukung operasi grafik dan interaktivitas. Beberapa *software* yang cukup populer untuk membuat simulasi antar lain Adobe Flash, Smart Apps Creator, Unity3D, Unreal Engine dan beberapa lainnya, sedangkan bahasa pemrograman yang secara umum digunakan untuk membangun sebuah simulasi pada umumnya adalah bahasa C, Javascript, HTML, Actionscript dan beberapa lainnya. Adobe Flash dan Actionscript misalnya pada 2 dekade terakhir dipakai secara luas karena mudahnya pengelolaan grafis dan interaktivitas, sedangkan pada 1 dekade terakhir perkembangan HTML 5 dengan fitur canvas menjadi standar baru pengembangan aplikasi berbasis web, dan popularitasnya mulai menggeser Adobe Flash/Animate.

HTML 5 adalah sebuah bahasa pemrograman untuk menstrukturkan dan menampilkan sebuah tampilan web (*World Wide Web*), yang merupakan elemen utama dari Internet. HTML 5 merupakan revisi kelima dari HTML yang mana HTML mulai dikembangkan pada tahun 1990. Tujuan utama pengembangan HTML5 adalah untuk memperbaiki teknologi HTML agar mendukung teknologi multimedia terbaru, mudah dibaca oleh manusia dan juga mudah dimengerti oleh mesin. Salah satu fitur yang didukung oleh HTML 5 yang dapat dimanfaatkan untuk memanipulasi grafis dan membentuk interaktivitas adalah `<canvas>`.

Canvas merupakan salah satu elemen dari HTML 5 yang memungkinkan untuk pengelolaan grafis dinamis menggunakan kode. Canvas divisualisasikan dalam bentuk area yang memiliki atribut lebar (*width*) dan tinggi (*height*), dapat dimanipulasi dengan cepat menggunakan bahasa pemrograman Javascript. Sebagaimana dengan kanvas kosong tempat pelukis bekerja, elemen `<canvas>` menyediakan area berbasis *bitmap* kepada pengguna untuk menggambar grafik menggunakan JavaScript. Area tersebut dapat digunakan untuk menggambar grafik, membuat komposisi foto, membuat animasi 2D dan 3D (digunakan dengan WebGL),

melakukan pemrosesan atau *rendering* video *real-time* dan dapat dijalankan di sebagian besar aplikasi *internet browser*.

Selama 2 dekade terakhir, penulis lebih sering menggunakan aplikasi populer Adobe Flash/Animate dalam membuat simulasi, media pembelajaran atau media lain yang melibatkan interaktivitas. Namun demikian, jika dibandingkan dengan Adobe Flash/Animate dalam hal kemampuan untuk membuat sebuah media interaktif, maka masing-masing memiliki kelebihan dan kekurangan. Berdasarkan pengalaman penulis, keduanya dapat dibandingkan sebagaimana tabel berikut :

Tabel 2.1 Perbandingan Adobe Flash dan HTML 5

Variabel	Adobe Flash/Animate	HTML 5 Canvas
Kebutuhan Software	Software berbayar (Adobe Flash/Animate), maupun software gratis (Flash Develop)	Dapat dibuat dengan menggunakan editor teks sederhana seperti notepad dan dijalankan langsung melalui browser (Chrome, Firefox, Opera, IE)
Bahasa Pemrograman	Actionscript 3	Javascript Keduanya memiliki kesamaan yang relatif tinggi.
Pengelolaan Grafis	Dilakukan secara langsung, <i>drag and drop</i> melalui <i>authoring tool</i> (Flash IDE) atau melalui pemrograman	Dilakukan melalui pemrograman. Meskipun demikian terdapat beberapa aplikasi tambahan yang mendukung pengelolaan grafis dengan metode <i>drag and drop</i>
Output	SWF Multiplatform (PC, Android, IOS)	File HTML yang dapat dijalankan pada browser. Melalui aplikasi tambahan juga dapat menghasilkan luaran <i>multiplatform</i> .
Dukungan referensi	Sangat banyak, mulai dari gratis sampai premium	Sangat banyak, berbasis komunitas dan sebagian besar gratis.
Kemudahan operasional	Relatif mudah	Tingkat menengah (membutuhkan pemahaman yang lebih tinggi)
Keamanan	Kurang aman, dan mulai ditinggalkan oleh sebagian besar internet browser.	Relatif aman, karena memiliki banyak pengaturan yang ketat dalam penggunaan data pengguna
Fleksibilitas	Fleksibel, memiliki banyak fitur yang siap untuk dipakai	Fleksibel, memiliki banyak dukungan dari komunitas yang menambahkan fitur-fitur tertentu melalui <i>code library</i> .

Jika anda menelusuri perbandingan antara Flash dan HTML5 Canvas, tentu saja anda akan mendapatkan hasil yang sangat banyak dan pada saat ini dukungan lebih condong ke HTML 5 Canvas. Secara personal saya sangat menyukai Adobe Flash dan telah membuat banyak aplikasi dengan Adobe Flash, namun dalam perkembangannya HTML 5 Canvas sangat dimungkinkan akan dan telah melampaui Flash, sehingga buku ini sengaja saya tulis dengan menggunakan HTML 5.

2.2 Struktur HTML Canvas

Sebelum pembahasan tentang canvas, kita perlu mengenal sedikit tentang standar HTML5 yang akan kita gunakan untuk membuat halaman simulasi. HTML adalah bahasa standar yang digunakan untuk membuat halaman di *World Wide Web*. Halaman HTML dasar dibagi menjadi beberapa bagian, umumnya <head> dan <body> yang seringkali disebut sebagai *tag*. Spesifikasi HTML5 baru menambahkan *tag* baru beberapa, seperti <nav>, <article>, <header>, dan <footer>.

Tag <head> biasanya berisi informasi yang akan digunakan oleh HTML seperti judul halaman, *meta data*, dan kode-kode tambahan. Sedangkan *tag* <body> untuk membuat halaman konten halaman HTML. Secara sederhana struktur kode HTML adalah sebagai berikut :

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.      <title>Judul Halaman</title>
5.  </head>
6.  <body>
7.      <p>Anda sedang belajar HTML</p>
8.  </body>
9. </html>
```

Kode di atas dapat dijalankan oleh internet browser dan akan menampilkan konten yang terdapat di antara *tag* <body>, yaitu tulisan “Anda sedang belajar HTML”.

Tag <canvas> merupakan fitur yang pertama kali dikenalkan oleh Apple pada tahun 2004, dan seiring waktu menjadi standar bagi *internet browser*. Seperti penjelasan sebelumnya *tag* <canvas> akan menjadikan konten halaman web dapat dimanipulasi dengan mengelola grafik dan interaktivitas. *Tag* <canvas> diletakkan di antara *tag* <body> dan perlu didefinisikan ukuran lebar dan tingginya. Perhatikan

struktur HTML untuk membentuk canvas dengan ukuran 800 x 600 *pixel* sebagai berikut:

```
<canvas id="scene" width="800" height="600"></canvas>
```

Setelah *tag* `<canvas>` tersebut dituliskan pada struktur HTML, maka canvas siap dimanipulasi dengan menggunakan kode Javascript.

2.3 Javascript dan Kode untuk Canvas

Javascript merupakan bahasa pemrograman tingkat tinggi, yang dapat berjalan hampir di semua internet browser. Javascript memiliki standar ECMAScript, yang berarti jika anda pernah belajar pemrograman dengan kode standar yang sama (contoh Actionscript 3.0 untuk Adobe Flash/Animate), maka anda akan menemukan kemiripan dan mudah untuk dipelajari. Namun, jika anda adalah pemula dalam pemrograman, maka anda juga tidak perlu khawatir karena bahasa Javascript sangat mudah untuk dipelajari.

Untuk memulai menuliskan kode Javascript maka diperlukan *tag* `<script>` yang bisa diletakkan di antara *tag* `<head>` maupun diantara *tag* `<body>`. Untuk lebih jelasnya perhatikan contoh membuat gambar garis pada canvas sebagai berikut :

A. Tutorial 1 – Memahami Javascript dan Canvas

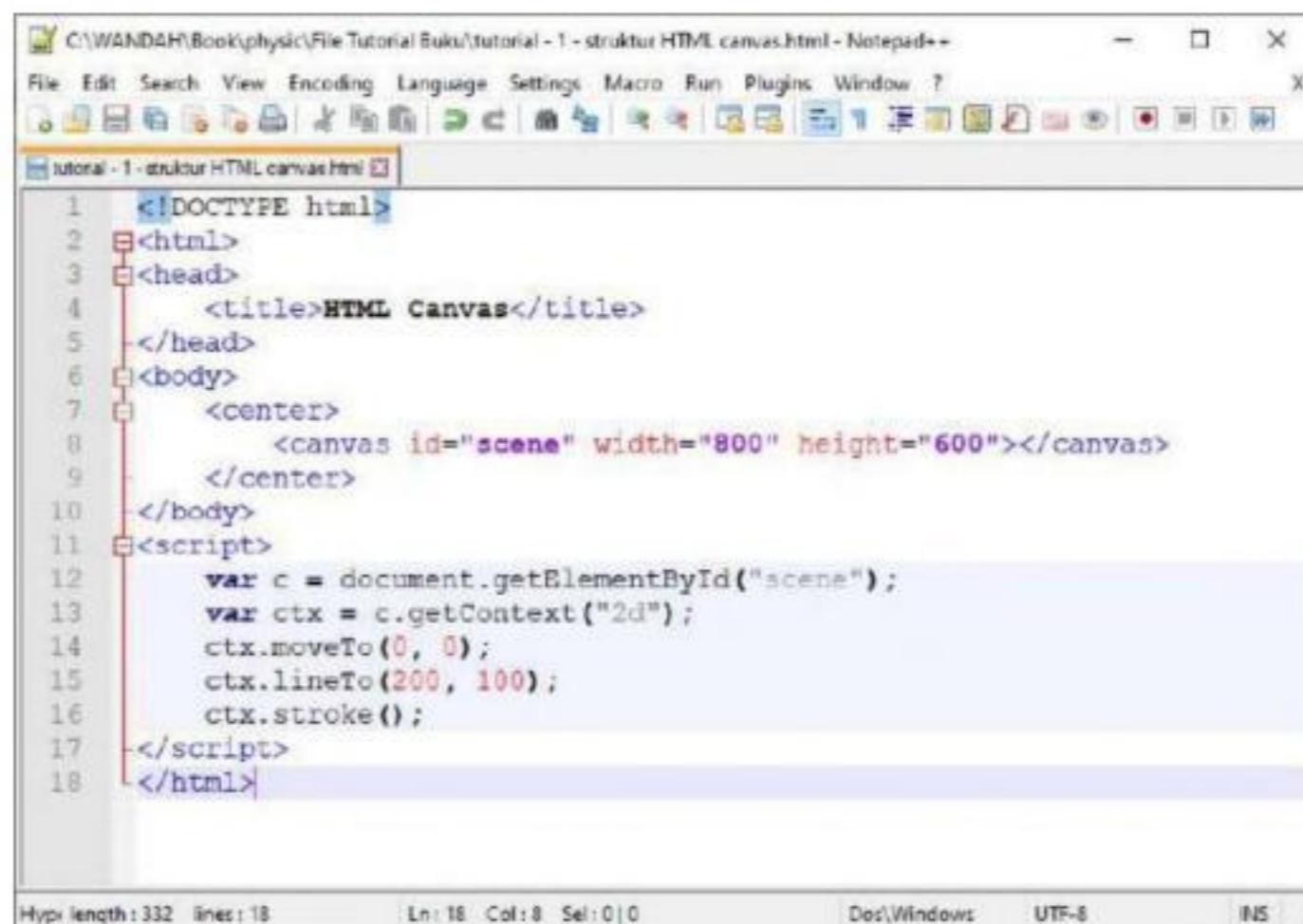
a. File HTML

1. Bukalah sebuah aplikasi editor teks seperti Notepad. Penulis menyarankan anda untuk menggunakan aplikasi **Notepad++**, karena mendukung sistem penulisan kode dengan struktur dan warna yang lebih baik daripada Notepad biasa.
2. Ketikan kode berikut :

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.      <title>HTML Canvas</title>
5.  </head>
6.  <body>
7.      <center>
8.          <canvas id="scene" width="800" height="600">
9.          </canvas>
10.     </center>
11.  </body>
12.  <script>
13.      var c = document.getElementById("scene");
14.      var ctx = c.getContext("2d");
```

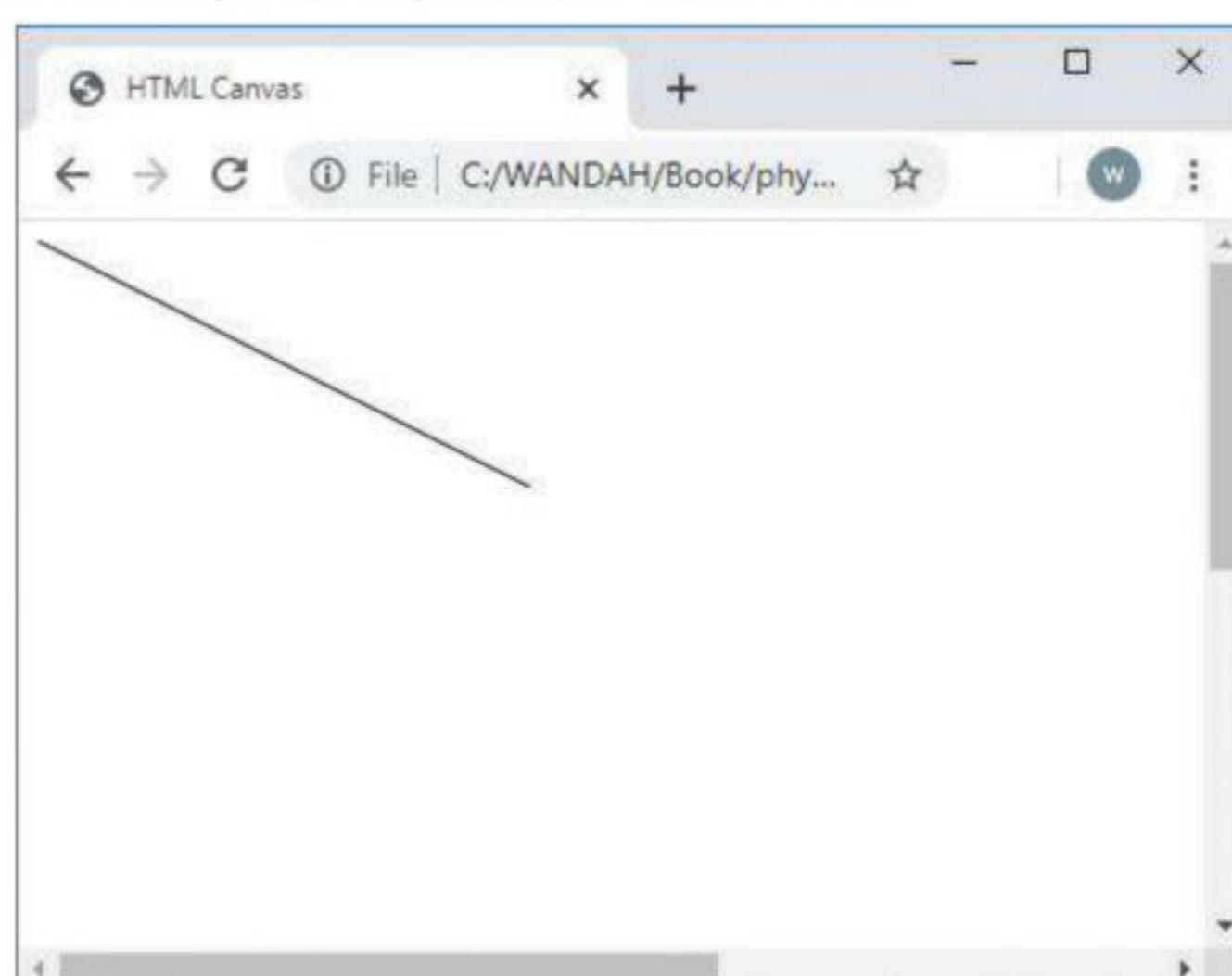
```
14.         ctx.moveTo(0, 0);
15.         ctx.lineTo(200, 100);
16.         ctx.stroke();
17.     </script>
18. </html>
```

3. Simpan dengan nama **tutorial-1.html**.



Gambar 2.1 Penulisan kode HTML dan JS di Notepad++

4. Dobel klik file tutorial-1.html tersebut, untuk membukanya dengan internet browser *default* di komputer anda. Atau jika anda ingin membukanya dengan internet browser lainnya, anda cukup mengklik kanan file dan pilih menu **open with..**.
5. Anda akan mendapati garis pada layar browser anda.



Gambar 2.2 Hasil tutorial 1, gambar garis pada canvas

b. Penjelasan Program

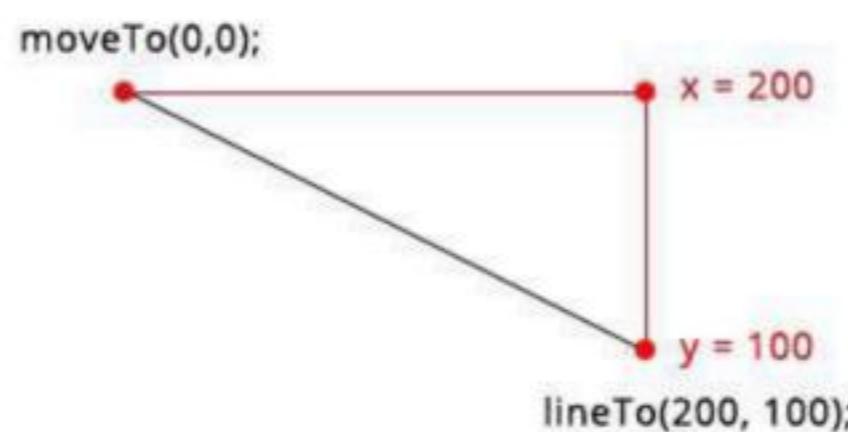
Pada tutorial 1 baris 8, canvas didefinisikan melalui tag <canvas> dengan dimensi 800x600 pixel. Secara default, canvas akan berwarna putih dan tidak memiliki garis tepi. Pada tahapan selanjutnya kita dapat merubah atribut dari canvas, memberikan warna, *background* dan elemen grafis lainnya.

Kode Javascript untuk memanipulasi canvas dituliskan di antara tag <script> ada baris 11-17.

```
12.     var c = document.getElementById("scene");
13.     var ctx = c.getContext("2d");
14.     ctx.moveTo(0, 0);
15.     ctx.lineTo(200, 100);
16.     ctx.stroke();
```

Pada baris 12, didefinisikan sebuah variabel `c` dengan mengambil `canvas` sebagai nilainya (kode `document.getElementById` akan mencari sebuah `tag` yang memiliki *id* tertentu, dalam hal ini `tag` yang memiliki *id* "scene" adalah `tag <canvas>`).

Pada baris 13, variabel `ctx` didefinisikan sebagai konteks (elemen dasar dari `canvas`). Setelah terdefinisikan, maka kita dapat memanipulasi konteks tersebut, dimulai dengan berpindah ke kordinat 0,0 dan menarik garis ke kordinat 200,100. Kode `stroke` pada baris 16 akan menampilkan sebuah garis sesuai dengan pengaturan pada 2 baris sebelumnya.



Gambar 2.3 Metode menggambar garis pada canvas

Jika anda perhatikan penulisan kode di tiap-tiap baris, terdapat sebuah pola atau keteraturan. Hal ini dalam dunia pemrograman disebut dengan *syntax* yaitu seperangkat aturan yang harus dipenuhi dalam penulisan kode. Selain *syntax* pada tutorial selanjutnya akan ditemukan istilah *variabel*, *function* dan sebagainya. Secara sederhana penjelasan dari istilah-istilah tersebut adalah sebagai berikut :

1. *Syntax*

Dalam pemrograman komputer sintaks berarti seperangkat aturan yang harus dipenuhi dalam penulisan kode. Sintaks meliputi simbol, kata, dan fungsi yang dapat digunakan serta bagaimana menggunakannya dalam sebuah struktur kode. Kesalahan umum yang sering dilakukan oleh programer adalah kesalahan sintaks (*Syntax error*), dimana kesalahan sintaks akan menyebabkan kode tidak bisa dijalankan oleh aplikasi. Berikut adalah aturan dalam penulisan Javascript (JS)

a) *Case Sensitive*

Penulisan kode JS menggunakan format *sensitive case*, sehingga setiap kode harus ditulis tepat sebagaimana mestinya. Satu perbedaan karakter akan menghasilkan perbedaan, sebagai contoh 2 baris kode berikut berbeda karena menggunakan karakter X besar dan x kecil.

```
1. var sumbuX;  
2. var sumbuX;
```

Untuk mempermudah penulisan kode maka digunakan metode *camelCase* yaitu menulis dua kata atau lebih secara sambung menyambung, dengan digunakan huruf besar pada awalan kata ke dua dan seterusnya. Misal

```
1. var namaVariabelSimulasi;
```

b) *Semicolons*

Semicolons (;) atau titik koma dalam JS dapat digunakan sebagai akhir dari sebuah baris kode. Apabila karakter ; tidak anda tuliskan maka, anda dapat mengganti baris dengan menggunakan enter.

```
1. kodeBaris1  
2. kodeBaris2;kodeBaris3;
```

c) *Parentheses*

Parentheses atau tanda kurung () digunakan untuk mengubah urutan dalam sebuah operasi/fungsi. Operasi yang diberikan tanda kurung akan dijalankan terlebih dahulu oleh program. Perhatikan contoh berikut :

```
1. var jarak = Math.sqrt((titikA.x - titikB.x)*( titikA.x -  
titikB.x)+(titikA.y - titikB.y)*( titikA.y - titikB.y));
```

Pada baris tersebut variabel *jarak* merupakan perhitungan *Phytagoras* yang secara umum digunakan untuk menghitung jarak antara *titikA* dan *titikB*. Keberadaan *parentheses* akan memulai operasi $(\text{titikA.x} - \text{titikB.x})$ terlebih dahulu, dilanjutkan dengan perkalian, dan pada akhirnya kode *Math.sqrt* akan menghitung akar dari kuadrat jarak.

Dalam pemrograman komputer, sistem perhitungan matematika akan dilakukan sesuai dengan peraturan yang ada seperti operasi perkalian akan didahulukan jika dibandingkan dengan penjumlahan atau pengurangan, dan operasi yang berada pada tanda kurung, akan dikerjakan terlebih dahulu. Untuk lebih jelasnya perhatikan contoh sederhana berikut :

```
1. console.log(1 + 2 * 3); // akan menghasilkan nilai 7  
2. console.log((1 + 2) * 3); // akan menghasilkan nilai 9
```

Parentheses juga dapat digunakan untuk memberikan masukan pada sebuah fungsi. Perhatikan contoh berikut :

```
1. tombolAktif = cekTombol(event);
```

Pada baris tersebut *tombolAktif* didefinisikan sebagai hasil dari fungsi *cekTombol*.

d) *Code blocks*

Dalam pemrograman JS, satu baris atau lebih yang terletak di dalam kurung kurawal `{ }` disebut sebagai satu blok kode. Satu blok kode merupakan sebuah paket yang biasanya digunakan dalam sebuah fungsi, *class*, *loop* dan sebagainya. Perhatikan baris berikut :

```
1. function akarKuadrat(num) {  
2.     //menghitung akar dari nilai num  
3.     return Math.sqrt(num);  
4. }
```

Dalam baris kode di atas fungsi akarKuadrat memiliki satu kode blok yaitu kode di baris 2-3.

e) *Whitespace*

Whitespace merupakan istilah untuk jarak penulisan, tab, enter dan spasi yang ditujukan untuk mempermudah penulisan kode. Perhatikan kode di atas, pada baris 2 dan 3 ditulis menjorok masuk, hal tersebut diistilahkan sebagai *whitespace* (ruang kosong). Pada kasus yang lain, programer akan memberikan jarak penulisan (satu kali enter) antar blok kode. *Whitespace* diabaikan oleh aplikasi saat mengkompilasi kode, namun sangat bermanfaat bagi programer untuk membantu mengecek struktur kode.

f) *Comments*

Comments atau komentar merupakan catatan yang bisa ditambahkan pada kode untuk mempermudah penjelasan kode. Ketika menulis kode dalam jumlah banyak (misal 2000 baris kode) dibutuhkan penjelasan-penjelasan pada baris kode untuk mempermudah proses pengecekan. Program secara otomatis akan mengabaikan baris komentar, sehingga kita tidak perlu khawatir dalam menggunakan komentar. Perhatikan baris 2 pada kode berikut :

```
1. function akarKuadrat(num) {  
2.     //menghitung akar dari nilai num  
3.     return Math.sqrt(num);  
4. }
```

Kata apapun yang berada di belakang karakter // akan dianggap sebagai komentar, dan tidak akan dijalankan oleh program. Apabila ingin menuliskan komentar lebih dari satu baris dapat menggunakan karakter /* **komentar** */

Penggunaan tanda /* */ pada tahapan selanjutnya akan sangat membantu programer untuk melakukan pengecekan kode. Beberapa baris kode yang diawali dengan karakter /* dan diakhiri dengan karakter */ tidak akan dijalankan oleh aplikasi, sehingga sangat berguna untuk mengetahui blok kode mana yang mengalami kesalahan.

2. Variable

Sistem kerja komputer adalah dengan menyimpan data dan mengolahnya. Dalam mengelola data komputer akan meyimpan sebuah nilai ke dalam memori, dimana seorang programer dapat mengatur nilai tersebut dan memberikannya nama. Nama dan nilai yang akan disimpan ke dalam memori itulah yang disebut sebagai **variabel**. Sebagai contoh, dibuat sebuah variabel untuk menghitung waktu, yaitu **var time = 0;** ini berarti komputer akan menyimpan sebuah data bernama **time** bernilai **0**. Variabel ini akan terus disimpan oleh memori komputer sepanjang aplikasi berjalan.

```
1. var score = 0;
```

3. Data Types

Data types atau jenis data adalah beberapa klasifikasi data yang digunakan dalam pemrograman JS. Secara mendasar jenis data di JS ada 6 yaitu *String, Number, int, uint, Boolean, dan Null*.

a) String

String adalah data yang bernilai tekstual dan penulisannya dalam JS menggunakan tanda petik, misalnya

```
var nama = "Izzan";
```

b) Number

Number adalah data yang bernilai angka tanpa batasan. Misal :

```
var nilaiUjian = 97;
```

c) Int

Int adalah kependekan dari integer yaitu data yang bernilai angka dari -2,147,483,648 sampai dengan 2,147,483,647 (32 bit data). Penggunaan variable bertipe *int* lebih efisien dari segi kecepatan pengolahan data dibanding dengan variabel bernilai *Number*.

d) Uint

Berbeda dengan *int* yang bernilai negatif sampai positif, *uint* hanya bernilai positif yaitu mulai dari angka 0 sampai dengan 4,294,967,295.

e) Boolean

Boolean merupakan nilai benar atau salah. Dalam JS bernilai *true* atau *false*.

f) Null

Null merupakan nilai kosong atau tidak bernilai, namun bukan tidak terdefinisikan (*undefined*).

Pada pengembangan berikutnya akan ditemukan tipe data yang lebih kompleks seperti *Array*, *Object* dan sebagainya.

4. Operators

Operators merupakan simbol yang digunakan untuk operasi matematika seperti penjumlahan (+), pengurangan (-), perkalian (*), pembagian (/), sama dengan (=) dan sebagainya. Pada contoh kode berikut digunakan operator penjumlahan.

```
1. score += 10;  
2. score = score + 10;
```

kedua baris kode di atas memiliki fungsi yang sama, yaitu nilai variabel score ditambah 10. Penulisan pada baris 1 lebih lazim dilakukan oleh programer, karena lebih menghemat karakter, yang berarti ukuran *file* akan menjadi lebih kecil.

5. Function

Function atau fungsi merupakan satu atau beberapa baris kode dalam satu blok, yang ditujukan untuk menyederhanakan suatu operasi dan dapat digunakan beberapa kali. Sebagai contoh, ketika membuat fungsi untuk menghasilkan akar kuadrat dari bilangan tertentu. Untuk menghitung akar kuadrat diperlukan sebuah nilai bilangan yang selanjutnya dengan kode dalam blok fungsi, bilangan tersebut dihitung akarnya. Perhatikan contoh berikut :

```
1. function akarKuadrat(num) {  
2.     //menghitung akar dari nilai num  
3.     return Math.sqrt(num);  
4. }
```

Untuk mengaktifkan fungsi tersebut cukup memanggil nama fungsi diikuti dengan nilai parameter yang ada.

```
5. var nilai = akarKuadrat(64); //akan menghasilkan nilai 8
```

6. Conditional if

Kondisi *if* merupakan logika dasar sebagian besar bahasa pemrograman. Dengan kode *if* sebuah kondisi dicek kebenarannya dan blok kode akan dieksekusi berdasarkan kondisi tersebut. Perhatikan baris berikut :

```
1. if (jarak < 10){ //jika jarak kurang dari 10 pixel
2.     kecepatan = 0;
3. }else{
4.     //jika jarak lebih dari 10 pixel kecepatan = 10
5.     kecepatan = 10;
6. }
```

Selain condisional *if*, juga terdapat beberapa logika dasar yang digunakan dalam pemrograman seperti *while*, dan *do..case*.

2.4 Debugging (Menguji Kode / Mencari Kesalahan Kode)

Dalam dunia pemrograman dikenal istilah *debugging* (menguji kode). Setiap programmer baik pemula maupun profesional dimungkinkan melakukan kesalahan penulisan kode (*syntax*), maupun kesalahan logika. Ketika kesalahan tersebut muncul, aplikasi tidak akan berjalan sempurna atau bahkan tidak berjalan sama sekali (menghasilkan layar yang kosong). Ketika hal ini terjadi kita tidak perlu panik, dan cukup melihat kesalahan yang terjadi pada panel konsole di internet browser.

Sebagai contoh ikuti langkah berikut :

A. Mencari Kesalahan Kode

1. Pada tutorial 1, baris ke 15 ubah kode *lineTo* menjadi *lineto* (huruf kecil semua).

menjadi

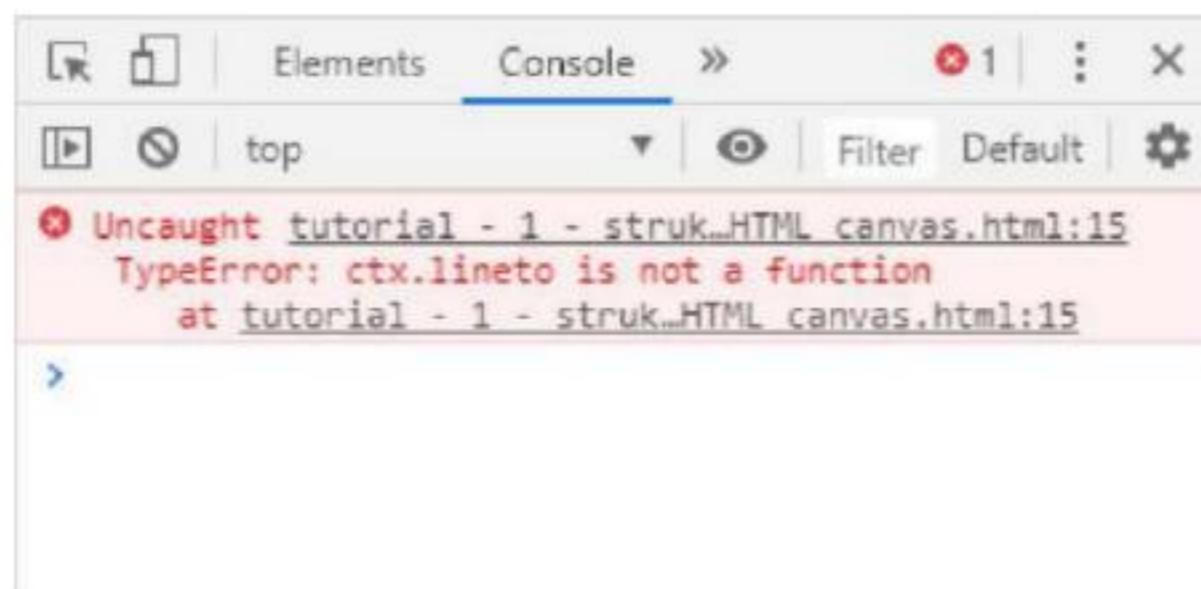
```
15.     ctx.lineTo(200, 100);
```

2. Simpan kembali *file* (dengan menekan Ctrl+S), lalu jalankan *file* melalui internet *browser*. Maka layar *browser* tidak akan menampilkan garis, sebagaimana pada tutorial 1 sebelum kita ganti baris 15 tersebut.
3. Untuk mengetahui kesalahan yang terjadi, anda dapat membuka menu *console* dengan *shortcut* sebagai berikut :

Mozilla Fire Fox	Ctrl+Shift+J
Google Chrome	Ctrl+Shift+J / F12
Internet Explorer	F12

Pada panel *console* tersebut akan muncul pesan *error* berwarna merah, yaitu :

Uncaught TypeError: ctx.lineto is not a function at tutorial - 1.html:15



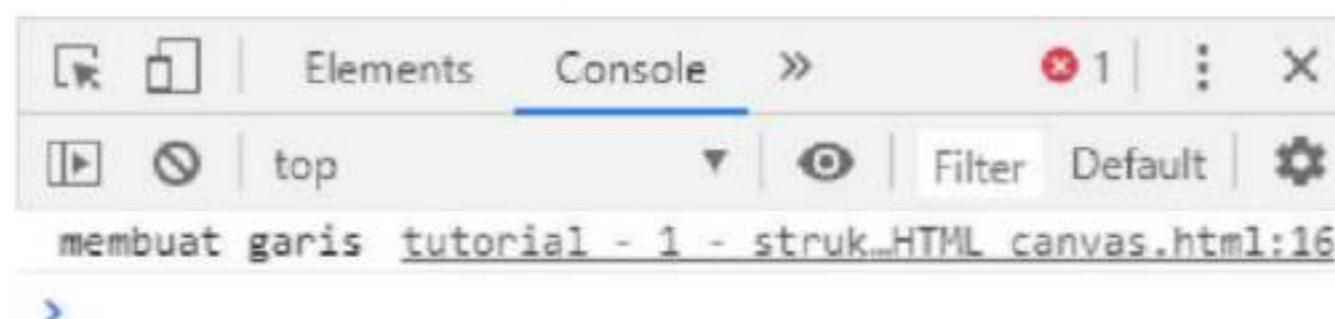
Gambar 2.4 Kesalahan yang dimunculkan pada panel *Console*

Dari panel *console* di atas, kita dapat mengetahui bahwa terdapat 1 kesalahan berada pada baris ke 15, yaitu `ctx.lineto` bukan sebuah fungsi. Dari pesan kesalahan ini kita dapat langsung mengedit kembali *file* tutorial-1.html, dan mengembalikan kode menjadi `lineTo`.

Selain memperingatkan akan adanya kesalahan, Javascript memiliki kode `console.log()` yang memungkinkan kita untuk mencatat pesan teks ke konsol Javascript untuk membantu menemukan masalah yang ada pada kode. Misal, pada baris 15 di bawah kode `ctx.lineTo`, ketikan kode berikut :

```
16.      console.log("membuat garis");
```

Simpan kembali *file*, kemudian buka pada *internet browser*. Kali ini bukalah panel *console*, dan anda akan menemukan pesan "membuat garis" pada *console*.



Gambar 2.5 Luaran pada panel *console*

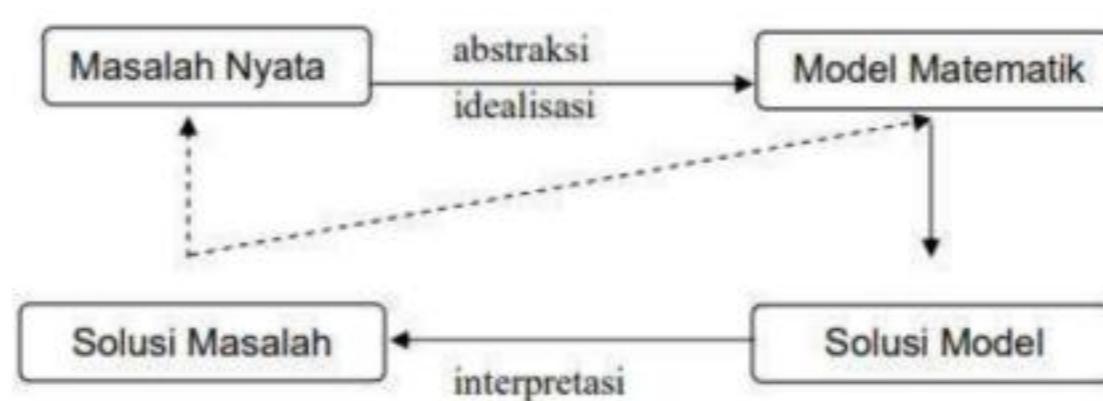
Fitur ini sangat berguna bagi programmer untuk mengetahui kinerja kode yang ditulisnya. Kita juga dapat menguji sebuah perhitungan, menguji jalannya logika dan melakukan pengujian lainnya melalui `console.log`. Setelah kode selesai diuji, kita dapat menghapus kode tersebut, atau memberikan komentar `//` di depannya agar tidak menampilkan pesan di panel *console*.

BAB 3

Pengembangan Simulasi dengan Javascript

3.1 Konsep Pengembangan Simulasi

Dalam mengadaptasi praktikum fisika menjadi bentuk simulasi, dapat digunakan model matematika. Model matematika mendeskripsikan suatu sistem dengan menggunakan pendekatan matematis. Pemodelan Matematika merupakan suatu proses merepresentasikan dan menjelaskan permasalahan pada dunia nyata ke dalam pernyataan^[7]. Melalui model matematika sebuah praktikum fisika diidentifikasi variabel-variabel yang terlibat di dalamnya, dibuat sebuah asumsi, dicari hubungan antar variabel dan asumsi, dan memformulasikan persamaan atau sistem persamaannya. Formulasi tersebut pada langkah selanjutnya diadaptasi ke dalam bentuk kode dalam pengembangan aplikasi.



Gambar 3.1 Pemodelan matematika

Model matematika akan menggambarkan suatu sistem dengan satu set variabel dan satu set persamaan yang membangun hubungan antara variabel. Variabel terdiri dari beberapa jenis; angka atau bilangan bulat, *string*, dan *boolean* (ya atau tidak). Sebagai contoh sederhana, hukum Newton 2 yang berbunyi "*Percepatan dari suatu benda akan sebanding dengan jumlah gaya (resultan gaya) yang bekerja pada benda tersebut dan berbanding terbalik dengan massanya*", maka dalam bentuk model matematika akan digunakan variabel percepatan (disimbolkan dengan a), resultan gaya (disimbolkan dengan F) dan massa benda (disimbolkan dengan m). Selanjutnya hubungan antar variabel berdasarkan bunyi hukum Newton tersebut dapat dituliskan dalam bentuk persamaan :

$$a = \frac{\sum F}{m} \text{ atau } \sum F = m \times a$$

Persamaan tersebut selanjutnya dapat diinterpretasikan ke dalam bentuk kode untuk membentuk simulasi. Sebagai contoh struktur kode untuk membentuk hukum Newton tersebut di atas adalah sebagai berikut :

```
1. var a = 2;  
2. var m = 10;  
3. var F = m*a;  
4. Console.log(F); //akan menghasilkan nilai F
```

Kemampuan menjabarkan sebuah praktikum fisika ke dalam bentuk model matematika sangat diperlukan untuk menghasilkan simulasi yang akurat. Simulasi ditulis dalam baris-baris kode dengan menggunakan model matematika yang telah ditentukan. Pada tahapan selanjutnya, pengembangan simulasi dapat dilakukan dengan tahapan-tahapan tertentu, yang mana dalam buku ini digunakan metode *waterfall*.

Konsep pengembangan aplikasi dalam bentuk simulasi yang digunakan pada buku ini menggunakan metode *waterfall* yang telah dimodifikasi untuk mempermudah pemahaman. Model *waterfall* adalah model pengembangan perangkat lunak yang menekankan fase-fase yang berurutan dan sistematis^[8] dimulai dari spesifikasi kebutuhan konsumen dan berkembang melalui proses perencanaan (*planning*), pemodelan (*modelling*), pengembangan (*construction*), dan distribusi (*deployment*), yang berujung pada dukungan terus menerus untuk sebuah perangkat lunak yang utuh (*maintainance*).

Secara detail metode *waterfall* termodifikasi yang dimaksud adalah sebagai berikut :

1. Perencanaan (*planing*)

Dalam buku ini, perencanaan simulasi dalam setiap bahasannya akan memilih satu jenis simulasi praktikum dan merencanakan dengan seksama tentang kebutuhan praktikum rill serta adaptasinya dalam bentuk virtual.

2. Pengembangan antarmuka (*graphic modelling*)

Dalam tahapan ini kebutuhan praktikum divisualisasikan dalam bentuk grafis. Grafis akan dibuat serepresentatif mungkin untuk mempermudah pemahaman pengguna. Tahapan ini akan diaktualisasikan dalam bentuk pengkodean grafis dasar menggunakan *library* yang telah disiapkan.

3. Penambahan Interaktivitas (*interactive construction*)

Setelah grafis untuk simulasi ditambahkan, maka tahapan ini secara spesifik akan menambahkan interaktivitas pada masing-masing komponen grafis untuk membentuk simulasi. Penambahan interaktivitas meliputi penambahan opsi untuk mengubah variabel-variabel praktikum, penampilan update grafis dan data praktikum, serta animasi atau gerakan objek-objek praktikum.

4. Distribusi (*deployment*)

Pada tahapan ini aplikasi akan dapat dijalankan melalui platform *web browser*, atau dengan pengaturan tambahan dapat didistribusikan pada *platform mobile* seperti Android atau IOS.

3.2 Tahapan Perencanaan

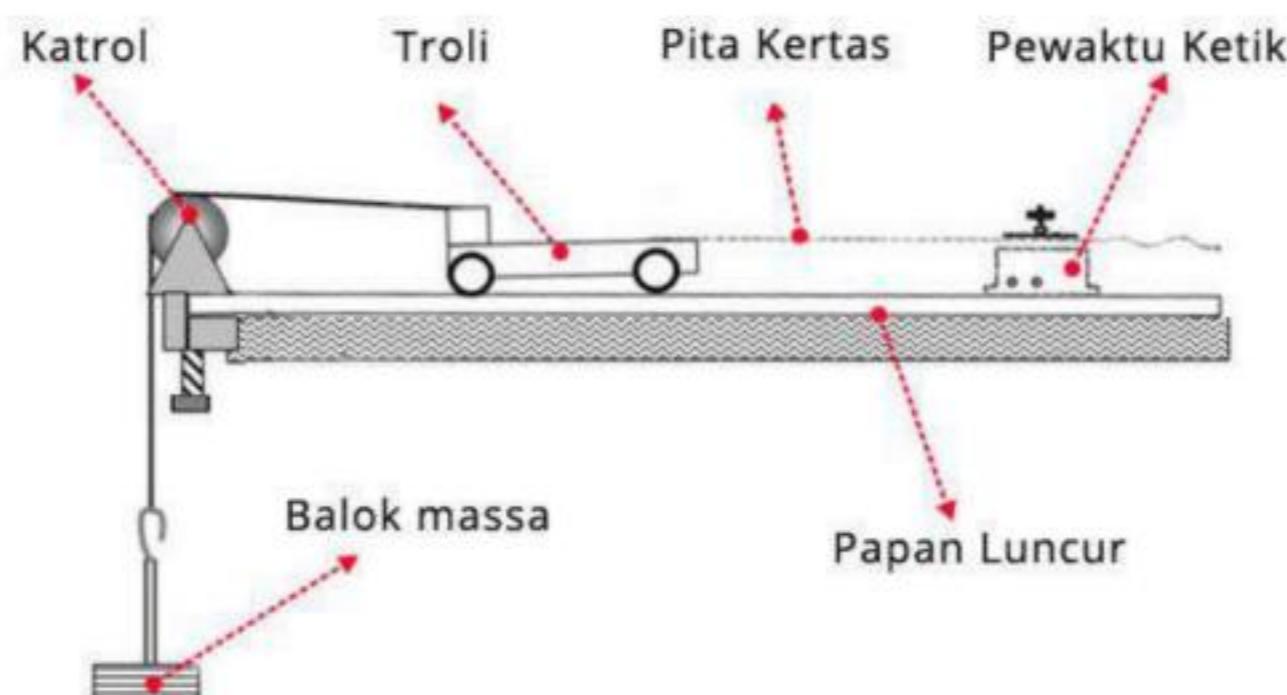
Sesuai dengan tahapan pengembangan aplikasi dengan metode *waterfall* di atas, pengembangan awal sebuah simulasi diawali dengan tahapan perencanaan. Tahapan perencanaan disini meliputi jenis praktikum apa yang akan disimulasikan berikut variabel-variabel terkait dalam praktikum tersebut. Sebagai contoh, kita akan mensimulasikan praktikum tentang gerak lurus berubah beraturan. Dalam praktikum yang sesungguhnya, digunakan beberapa alat seperti kereta luncur (troli), papan luncur, balok massa (beban), tali, serta pewaktu ketik (*ticker*).



Gambar 3.2 Praktikum gerak lurus berubah beraturan

Dalam percobaan di atas, beberapa variabel yang digunakan dalam praktikum antara lain adalah massa beban, massa troli, waktu, gaya gravitasi dan kecepatan. Beberapa variabel ini nantinya harus dapat diamati oleh pengguna aplikasi, agar tujuan dari praktikum virtual dapat tercapai sebagaimana pada percobaan yang sesungguhnya di laboratorium.

Setelah penentuan model praktikum dan variabel yang terlibat, maka perencanaan dilanjutkan dengan pembuatan visualisasi skematis agar praktikum tersebut lebih mudah divisualisasikan ke media 2 dimensional, yaitu layar *display* aplikasi. Maka secara sederhana, kita dapat mengadaptasi skema praktikum yang secara umum sudah terdapat pada buku pelajaran fisika. Sebagai contoh, praktikum gerak lurus berubah beraturan tersebut dapat digambarkan dengan skema sebagai berikut :



Gambar 3.3 Skema praktikum gerak lurus berubah beraturan

Pada praktikum tersebut variabel yang terlibat dapat dituliskan dalam rumus gerak lurus berubah beraturan (GLBB) sebagai berikut :

$$\Delta x = V_o \cdot t \pm \frac{1}{2} a t^2$$

Dimana Δx adalah jarak, V_o adalah kecepatan awal, t adalah waktu, dan a adalah percepatan. Rumus tersebut akan kita terapkan pada troli dengan variabel posisi ($x1$), maka variabel-variabel tersebut dalam bentuk kode untuk membentuk simulasi adalah sebagai berikut :

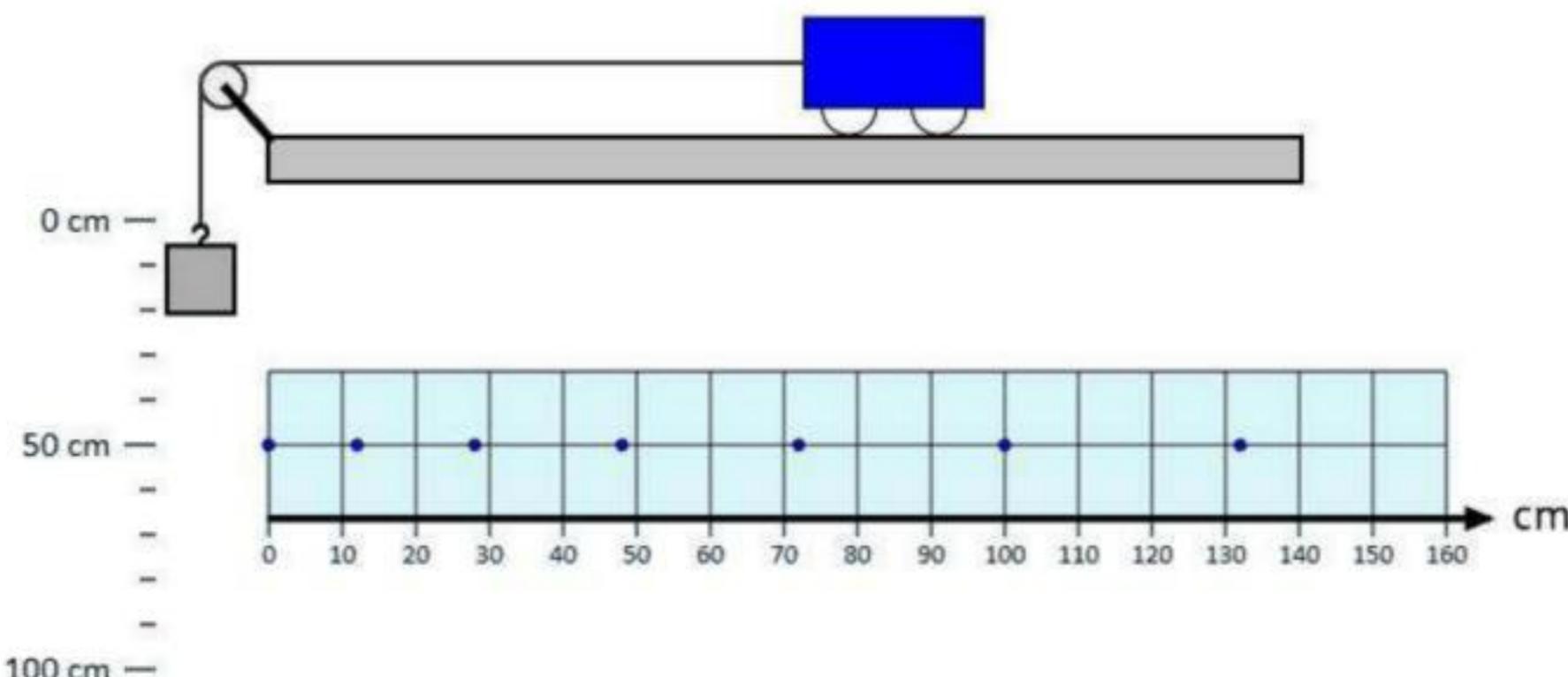
```
1. var x1, v, t, a;
2. x1 = v*t + 0.5*a*t*t;
3. troli.x = x1; //memposisikan troli sesuai variabel x1
```

Ketika masing-masing variabel (v , t , dan a) diatur nilainya sesuai dari input pengguna, maka variabel $x1$ akan berubah dan simulasi akan berjalan. Dalam contoh ini, perencanaan variabel dan penggunaannya juga harus ditentukan pada awal pengembangan simulasi untuk mempermudah proses selanjutnya.

3.3 Tahapan Pengembangan Antarmuka (*Graphic Modelling*)

Pada tahapan ini, skema yang sudah dibuat pada tahapan sebelumnya, divisualisasikan dalam bentuk gambar, dengan pengaturan tata letak (layout) yang baik sehingga memberikan pengalaman pengguna (*user experience*) yang baik.

Adaptasi praktikum yang sesungguhnya ke dalam bentuk virtual, membutuhkan adaptasi antarmuka yang baik. Sebagai contoh, pada praktikum di atas, terdapat pewaktu ketik (*ticker*) yang mana pada praktikum yang sesungguhnya akan menggambarkan titik-titik berjarak pada pita kertas. Dalam praktikum yang sesungguhnya, pita kertas dapat dilihat (dibaca) dari pandangan atas, sementara pada skema tersebut, pita kertas pada pandangan samping, hal ini yang harus dipikirkan oleh seorang pengembang simulasi virtual. Subtitusi pita kertas pada model praktikum virtual harus diwujudkan, tanpa mengubah tujuan utama pita kertas dalam praktikum riil. Dalam kasus ini, kita dapat mensubtitusi pita kertas dengan grafis yang menunjukkan pola perubahan kecepatan troli terhadap jarak dan waktu tempuh. Pada akhirnya desain antarmuka simulasi praktikum tersebut dapat digambarkan sebagai berikut :



Gambar 3.4 Antarmuka percobaan gerak lurus berubah beraturan

Tantangan berikutnya dalam pembuatan simulasi adalah menggambarkan objek ke layar aplikasi. HTML Canvas dengan Javascript memiliki banyak fitur untuk mengelola gambar, namun mewujudkan antarmuka di atas diperlukan beberapa puluh baris kode, yang mana bagi pemula hal ini sulit untuk diwujudkan. Oleh karena itu pada buku ini, proses pengembangan grafis simulasi akan menggunakan *library* khusus.

A. Javascript *Library*

Dalam mengembangkan sebuah aplikasi, seringkali developer membuat serangkaian fungsi-fungsi tertentu untuk mempermudah proses pengembangan. Serangkaian fungsi tersebut menggabungkan beberapa fungsi dan menyederhanakannya, serta

dapat dipakai berulang kali untuk proyek yang berbeda-beda. Inilah yang disebut sebagai Javascript *library*.

Sebagai bahasa pemrograman yang sangat populer, terdapat begitu banyak *library* maupun *framework* berbasis Javascript yang bisa kita gunakan. *Library* untuk grafis canvas misalnya, terdapat berbagai *library* yang cukup banyak dipakai seperti d3.js, chart.js, raphael.js, paper.js dan sebagainya. Tingkat kerumitan penggunaannya cukup beragam, ada yang dapat digunakan secara langsung dengan memanggil *file js* eksternal, dan ada yang menggunakan pengaturan yang lebih kompleks seperti pengaturan *server*, penggunaan *node*, dan sebagainya.

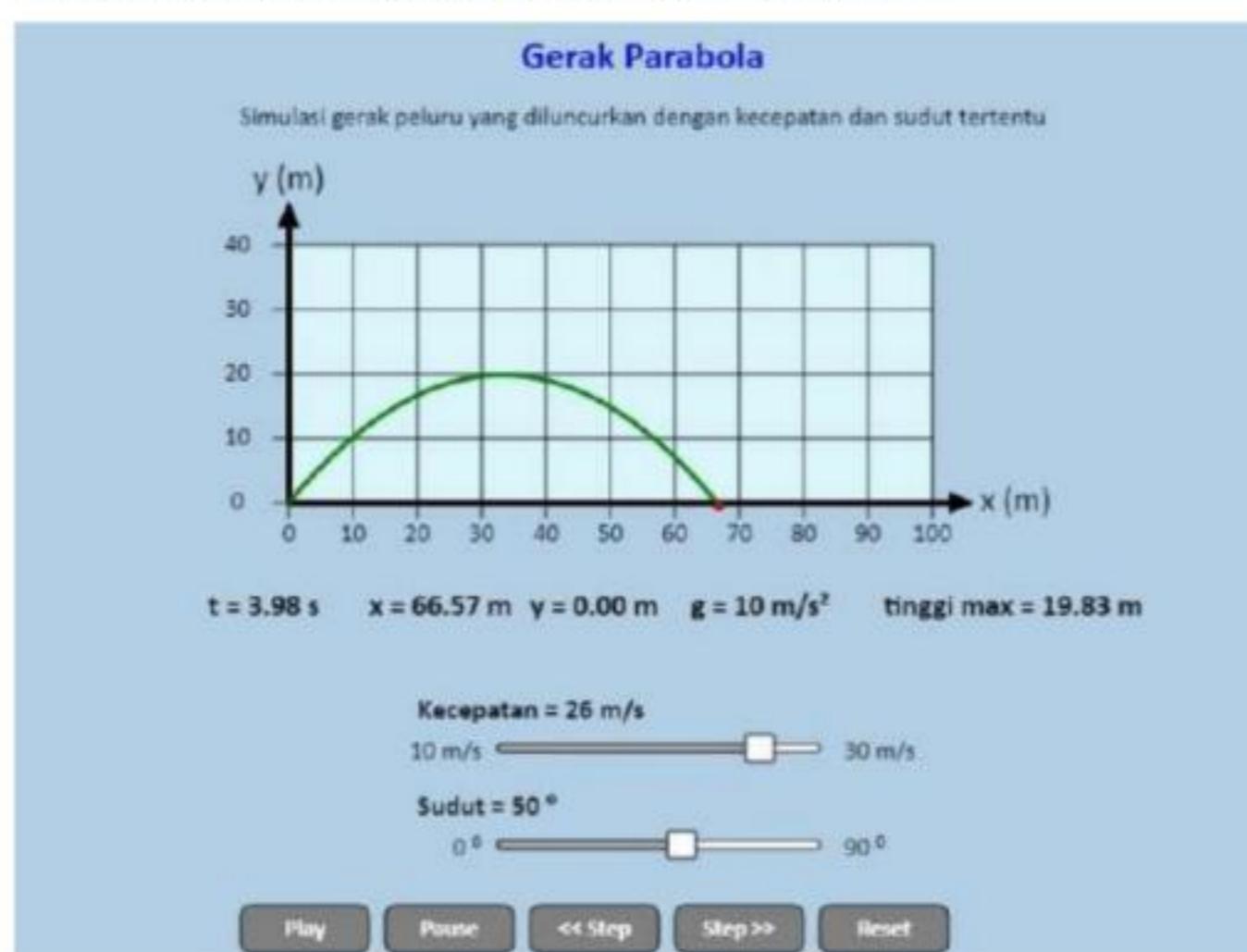
Mengingat buku ini ditujukan untuk pemula, maka *library* yang digunakan dalam buku ini adalah *library* yang saya tulis untuk menyederhanakan beberapa fungsi standar, dan dapat digunakan secara langsung dengan memanggil *file* eksternal bernama `simulasiDasar.js`. Untuk mengaktifkannya anda cukup meletakkan *file* `simulasiDasar.js` pada *folder* tempat anda mengerjakan simulasi, dan memanggilnya dengan tag `<script>`. Penjelasan lebih detail tentang penggunaan *library* ini, akan anda dapatkan pada sub bab selanjutnya.

3.4 Tahapan Pengembangan Interaktivitas (*Interactive Construction*)

Interaktivitas berarti hubungan dua arah antara pengguna dengan aplikasi (*feed* dan *feedback*). Dalam simulasi, interaktivitas memungkinkan pengguna untuk mengatur variabel-variabel yang ada, dan aplikasi akan meresponnya dengan melakukan perhitungan ulang terhadap variabel-variabel dinamis tersebut. Interaktivitas merupakan bentuk adaptasi utama dalam sebuah simulasi, karena melalui interaktivitaslah pengguna dapat mengatur jalannya simulasi dengan memberi masukan ke dalam aplikasi, mengetahui efek dari perubahan variabel terhadap jalannya simulasi melalui umpan balik yang ditampilkan oleh simulasi.

Elemen interaktivitas dasar yang akan digunakan di dalam buku ini antara lain tombol pengaturan simulasi, *slider* untuk mengubah nilai variabel, input teks untuk memasukkan nilai tertentu. Sebagai contoh dalam simulasi gerak parabola, terdapat beberapa tombol untuk mengatur jalannya simulasi seperti tombol *Play*, *Pause*, *Step*

Forward, Step Backward dan *Reset*, serta 2 buah *slider* untuk mengatur variabel yang terlibat dalam gerak parabola, yaitu sudut dan kecepatan.



Gambar 3.5 Elemen interaktif pada simulasi gerak parabola

3.5 Tahapan Distribusi (*Deployment*)

Pada tahapan ini aplikasi akan didistribusikan kepada pengguna. Distribusi simulasi yang dibuat menggunakan teknik HTML Canvas dengan Javascript sangat fleksibel untuk didistribusikan melalui berbagai *platform*. Sebagai contoh, untuk *file* HTML yang telah dibuat, kita cukup menguploadnya ke *server* dan *file* secara otomatis dapat dijalankan melalui *internet browser*. Untuk *platform* lain seperti Android atau iOS, kita dapat menggunakan aplikasi khusus untuk meng"*compile*" kode yang kita buat untuk menjadi *file* aplikasi siap pakai.

3.6 Sumber Belajar Javascript

Pada era internet ini, untuk mendapatkan sumber belajar kita tidak akan menemui kendala. Internet telah menyediakan berbagai sumber belajar, selama kita mampu mengoperasikan mesin mencari dengan baik. Dalam mengembangkan laboratorium virtual berbasis canvas, kita dapat menemukan tutorial Javascript pada *link-link* berikut :

1. <https://www.w3schools.com/js/>

Tutorial w3schools sangat mudah untuk dipelajari dan ditampilkan secara bertahap. Dengan mengikuti tutorial dari awal selangkah demi selangkah,

kita dapat mempelajari pemrograman dasar javascript. Khusus untuk canvas anda dapat membuka link :

https://www.w3schools.com/html/html5_canvas.asp

2. <https://javascript.info/>

Seperti halnya w3schools, tutorial pada javascript.info memiliki list kode javascript dan contoh penggunaannya. Pada situs ini kita bisa memilih kode apa yang ingin dipelajari.

3. <https://www.petanikode.com/tutorial/javascript/>

Bagi yang menginginkan tutorial berbahasa Indonesia, web petanikode merupakan salah satu alternatif terbaik dalam belajar bahasa pemrograman, khususnya Javascript.

4. <http://www.html5canvastutorials.com/>

Situs tersebut mengkhususkan pada tutorial berbasis html 5 canvas disertai dengan contoh-contoh aplikasi yang dikembangkan dengan javascript.

5. Situs lain atau buku elektorik yang tersedia secara *online*

BAB 4

Grafis Dasar

4.1 Menggunakan File *Library* simulasiDasar.js

Sesuai dengan penjelasan pada bab sebelumnya, membuat grafis untuk visualisasi simulasi praktikum dalam laboratorium virtual membutuhkan beberapa baris kode yang cukup sulit untuk dipraktikkan bagi pemula. Oleh karena itu beberapa baris kode disederhanakan dalam sebuah fungsi yang disebut dengan *library*, dalam buku ini *library* tersebut diberi nama **simulasiDasar.js**.

Untuk memanfaatkan *library* simulasiDasar.js, anda membutuhkan sebuah *file library* yang dapat anda unduh melalui situs www.wandah.org/js/simulasiDasar.js. Adapun pengaturan *file* dalam setiap proyek simulasi yang akan dibuat adalah sebagai berikut :

1. Sebuah *folder* khusus untuk menampung semua *file* proyek simulasi.
2. Sebuah *file* HTML yang merupakan *file* HTML utama dari simulasi
3. *File library* simulasiDasar.js.
4. *File* simulasi.js yang merupakan kode Javascript dari simulasi. *File* ini nantinya akan di"*load*" bersamaan dengan *library* simulasiDasar.js. Penamaan *file* ini nantinya akan diberikan penomoran yang sesuai dengan masing-masing tutorial agar mempermudah identifikasi. Misal, pada tutorial 2 akan digunakan nama simulasi-2.js.
5. Sebuah *folder images*, untuk menampung *file* gambar (*bitmap*) yang akan digunakan dalam simulasi.
6. *Library tambahan*, jika anda membutuhkan *library* pendukung lainnya. Dalam buku ini, tidak digunakan *library* tambahan lainnya untuk mempermudah penjelasan.

Name	Date modified	Type
images	03/05/2020 10.31	File folder
resultan vektor	03/04/2020 22.16	Chrome HTML Document
simulasi-1	15/04/2020 11.41	JS File
simulasiDasar	23/04/2020 07.36	JS File

Gambar 4.1 Struktur *file* dan *folder* proyek simulasi

4.2 Struktur utama file HTML

Pada *file* tutorial 1 pada Bab 2, telah dijelaskan penggunaan fitur canvas. Pada tahapan selanjutnya, struktur *file* HTML yang digunakan di dalam buku ini relatif tetap, dan hanya perlu diubah nama *filenya* saja. Adapun struktur *file* yang dimaksud adalah sebagai berikut :

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.      <title>Judul Simulasi</title>
5.      <script src="simulasiDasar.js"></script>
6.  </head>
7.  <body>
8.      <center>
9.          <canvas id="scene" width="800" height="600">
    </canvas>
10.     </center>
11.     <script src="simulasi-1.js"></script>
12. </body>
13. </html>
```

Untuk penggunaan library secara *online*, anda dapat menggunakan kode pada baris 5 sebagai berikut :

```
5.  <script
src="https://www.wandah.org/js/simulasiDasar.js">
</script>
```

Pada baris 5, *file library* **simulasiDasar.js** dipanggil, yang selanjutnya diikuti dengan *file* **simulasi-1.js** pada baris 12 yang merupakan *file* simulasi utama dalam membentuk simulasi yang diinginkan.



Untuk mempermudah pemahaman anda, ikutilah tutorial secara bertahap, dimulai dari tutorial menampilkan grafis.

4.3 Membuat grafis dasar

Tutorial kedua pada buku ini akan menjelaskan fungsi dasar dalam membuat grafis pada kanvas. Perhatikan langkah-langkah dan penjelasan pada masing-masing tutorial untuk mempermudah pemahaman anda.

A. Tutorial 2a – Membuat Garis

a. *File* HTML

1. Buatlah sebuah *folder* khusus dengan nama **tutorial-2**. Folder ini digunakan untuk menyimpan seluruh *file*.

2. Bukalah aplikasi **Notepad++** atau aplikasi teks *editor* lainnya.
3. Ketikkan kode HTML utama di atas. Ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
6.      <script src="simulasi-2.js"></script>
```

4. Simpan file HTML tersebut dengan nama **tutorial-2.html**. Pada **Notepad++** anda harus mengetikkan nama file berikut ekstensinya (.html), karena jika tidak ditambah ekstensi, secara *default* file akan disimpan dalam format TXT. Langkah ini akan dilakukan pada tiap-tiap tutorial, kita cukup mengubah nama *file* simulasi yang dipanggil dan nama *file* html.

b. *File library*

1. *Copy paste file* **simulasiDasar.js** ke dalam folder **tutorial-2**.

Langkah ini bertujuan agar file *library* dipanggil secara langsung dari *folder* yang sama. Metode ini mudah untuk dilakukan oleh pemula, karena tidak membutuhkan pengaturan yang rumit, cukup *copy paste file* saja. Pada mode yang lebih profesional, *file library* akan dipanggil dari server tertentu menggunakan CDN (*Content Delivery Network*) dan sejenisnya.

c. *File simulasi-2.js*

1. Pada Notepad++, buatlah sebuah *file* baru.

2. Ketikan kode berikut :

```
1. aturCanvas();  
2. setJudul("Grafis dasar HTML Canvas");  
3.  
4. function setSimulasi(){  
5.     //menambahkan background warna  
6.     hapusLayar("#e8e8e8");  
7.     //membuat garis  
8.     garis(80, 100, 250, 100);  
9.     garis(80, 120, 350, 120, 4, "blue");  
10.    garis(80, 140, 450, 140, 3, "red", "dash");  
11.    garis(80, 160, 450, 160, 2, "#db2191", "dash-10-5");  
12. }  
13. setSimulasi();
```

3. Simpan dengan nama **simulasi-2.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-2.

d. Menjalankan tutorial

Untuk menjalankan tutorial, dobel klik file **tutorial-2.html**, untuk membukanya dengan *internet browser default* di komputer anda. Atau jika anda ingin membukanya dengan *internet browser* lainnya, anda cukup mengklik kanan *file* dan pilih menu **open with..**.



Gambar 4.2 Hasil tutorial membuat garis

e. Penjelasan Program

Untuk memulai simulasi digunakan kode `aturCanvas`, adapun struktur kode ini adalah :

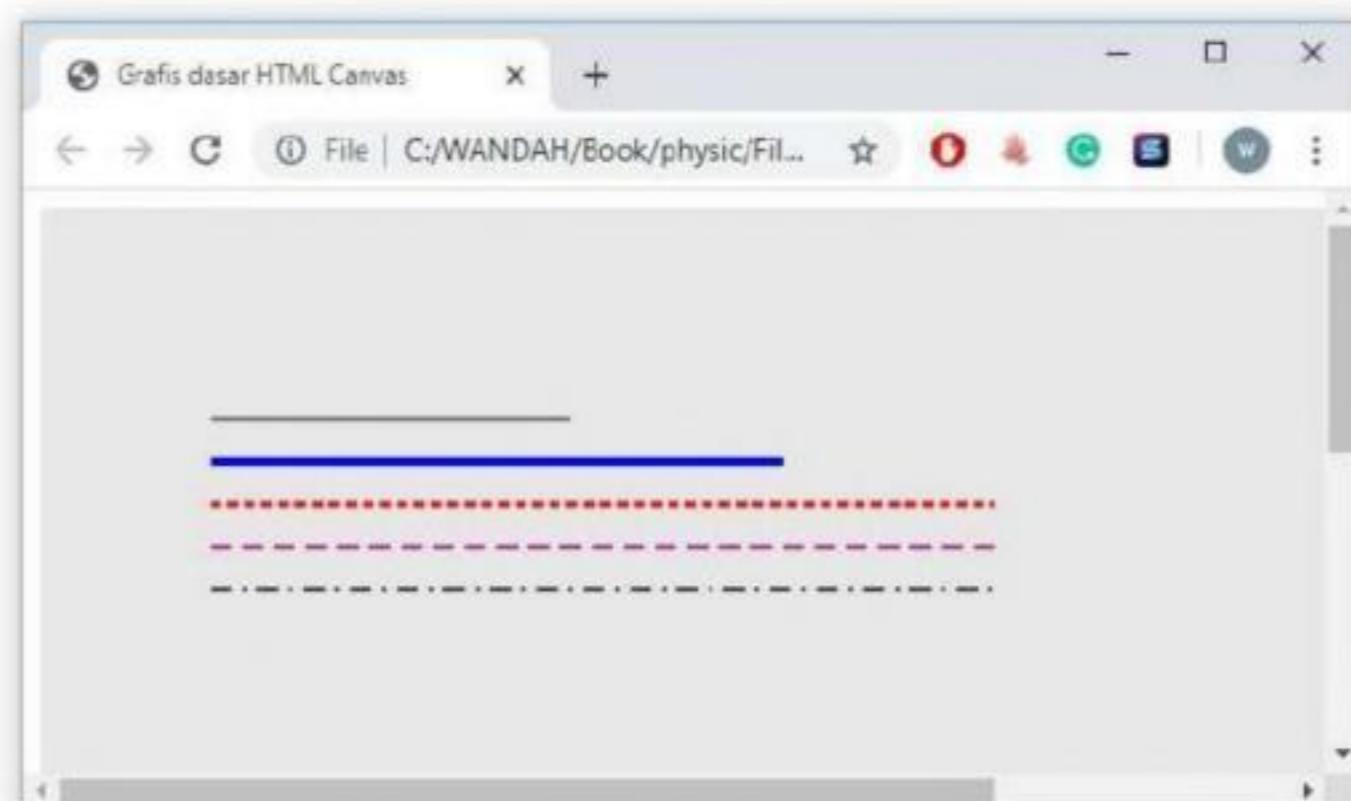
```
1. aturCanvas();
```

Fungsi `aturCanvas` digunakan untuk mengatur elemen `canvas` dan harus dieksekusi di awal program.

Pada baris 2 dituliskan kode

```
2. setJudul("Grafis dasar HTML Canvas");
```

Kode ini berfungsi untuk mengatur *tag <title>* pada *file HTML*, sehingga pada *tag <title>* berubah secara dinamis sesuai dengan *string* yang diinputkan. Judul halaman ini akan muncul pada panel di *internet browser* anda.

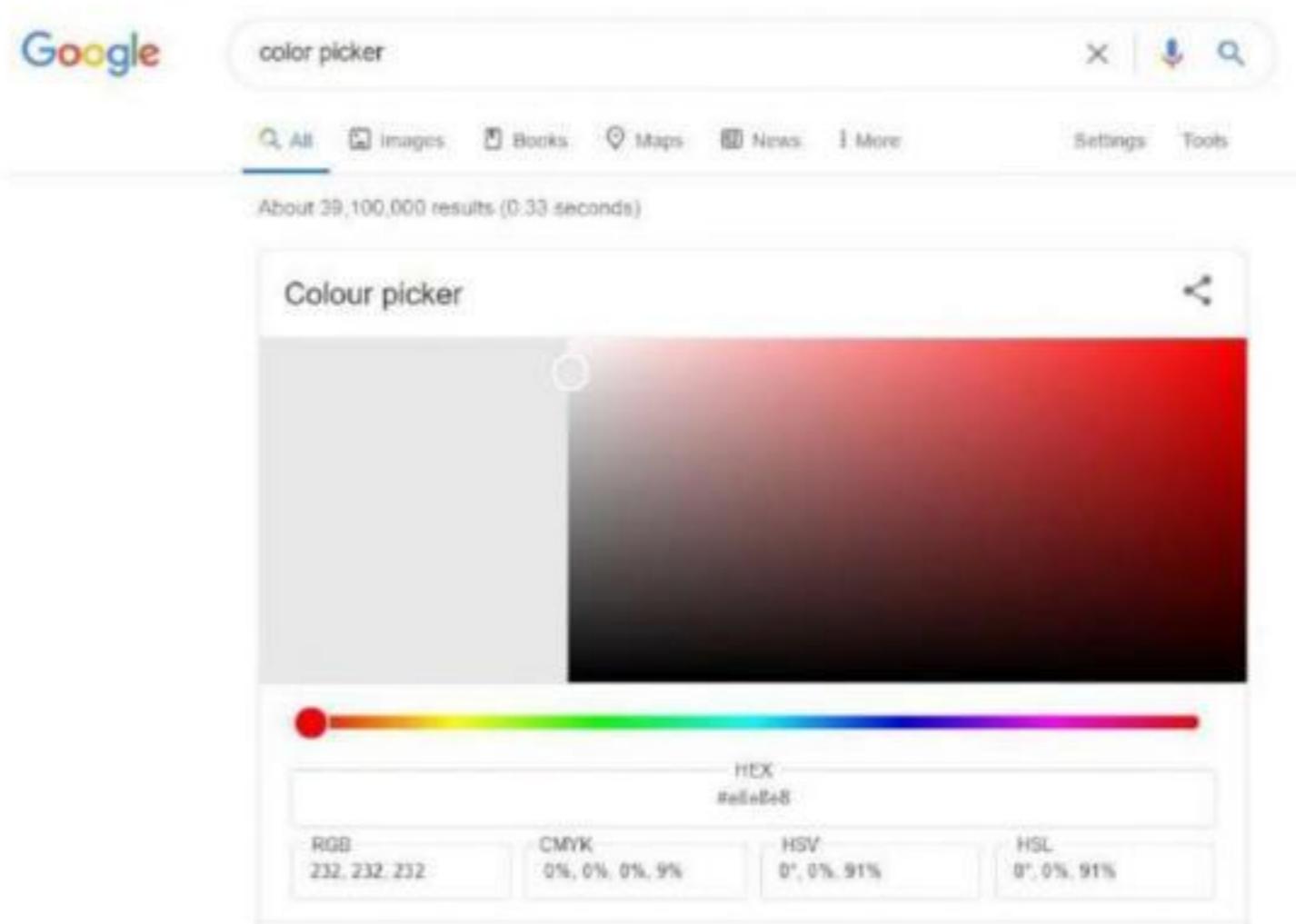


Gambar 4.3 Judul pada panel *internet browser*

Pada baris selanjutnya, didefinisikan fungsi **setSimulasi**, dengan beberapa perintah sebagai berikut:

```
5. hapusLayar("#e8e8e8");
```

Sesuai dengan namanya, fungsi `hapusLayar` digunakan untuk menghapus keseluruhan canvas dengan warna tertentu, dalam hal ini warna yang dimaksud adalah warna dalam kode *heksa* **#e8e8e8** atau warna abu-abu. Untuk memperoleh kode warna dengan mudah, anda dapat membuka situs google dan mengetikkan “*color picker*” pada kolom *search*. Kode heksa dapat anda dapatkan secara langsung pada layar seiring dengan pilihan warna yang anda tentukan.



Gambar 4.4 Menentukan kode warna dengan *color picker*

Untuk membuat garis digunakan struktur fungsi sebagai berikut :

```
garis(x1, y1, x2, y2, tebal, warna, tipe);
```

Pada fungsi `garis` di atas `x1, y1` adalah titik koordinat awal dari garis. `x2, y2` adalah titik koordinat akhir dari garis. Parameter `tebal`, digunakan untuk mengatur ketebalan garis dan secara default bernilai 1. Parameter `warna` digunakan untuk mengatur warna garis dan secara *default* bernilai hitam (“*black*”). Untuk parameter `warna`, anda dapat mengisinya dengan kode warna umum (baris 8 dan 9) atau mengisinya dengan kode heksa, sebagaimana pada kode baris 10.

Untuk tambahan jenis garis seperti garis putus-putus digunakan opsi “*dash*”. Opsi ini secara default akan menghasilkan garis putus-putus sepanjang 5 *pixel* dengan jarak 3 *pixel*. Anda dapat melakukan pengaturan yang lebih kompleks seperti “*dash-10-5-2-5*”, untuk menghasilkan garis 10 *pixel* diikuti jarak 5 *pixel*, dilanjutkan dengan garis 2 *pixel* dan jarak 5 *pixel*, sehingga menghasilkan garis seperti pada gambar berikut:



Gambar 4.5 Hasil pengaturan *dash-10-5-2-5*

Selanjutnya fungsi `setSimulasi` dipanggil pada baris 13 untuk menampilkan seluruh kode yang ada pada baris 5-11.

B. Tutorial 2b – Membuat Kotak

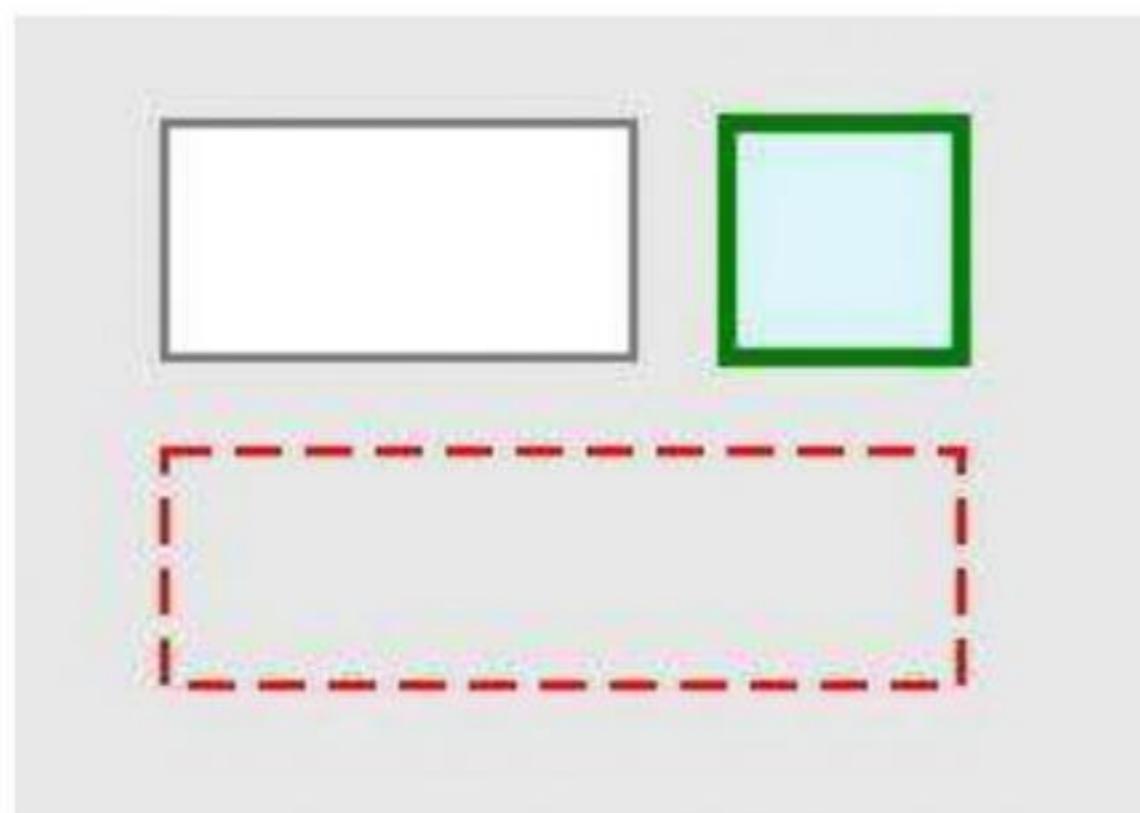
a. *File simulasi-2.js*

Untuk menjelaskan proses pembuatan sebuah kotak, pada tutorial ini sengaja tidak membuat proyek baru, namun melanjutkan tutorial sebelumnya. Perubahan yang akan dilakukan hanyalah penambahan baris pada kode `simulasi-2.js`. Untuk lebih jelasnya perhatikan langkah berikut :

1. Buka kembali *file simulasi-2.js* pada teks *editor Notepad++*.
2. Klik baris 12 tepat di depan karakter `}`, kemudian tekan **Enter**. Klik kembali baris 12, kemudian sisipkan kode berikut :

```
4. function setSimulasi() {
5.     //menambahkan background warna
6.     ....
12.    //membuat kotak
13.    kotak(500, 100, 100, 50);
14.    kotak(620, 100, 50, 50, 4, "green", "#daf6fb");
15.    kotak(500, 170, 170, 50, 2, "red", "none",
16.        "dash-10-5");
```

3. Simpan *file*.
4. Jalankan kembali *file tutorial-2.html* dengan membukanya pada *internet browser*, atau tekan tombol *refresh* (F5) apabila *file* sudah terbuka di *internet browser*. Kode di atas akan menghasilkan 3 buah kotak sebagai berikut :



Gambar 4.6 Hasil tutorial membuat kotak

b. Penjelasan Program

Untuk membuat kotak digunakan struktur fungsi sebagai berikut :

```
kotak(x, y, p, l, tebal, warna garis, warna isi, tipe);
```

Pada fungsi `kotak` di atas `x, y` adalah titik koordinat awal dari kotak. `p` adalah panjang dalam satuan *pixel*. `l` merupakan lebar dari kotak. Parameter `tebal`, digunakan untuk mengatur ketebalan garis dan secara *default* bernilai 1. Parameter `warna garis` digunakan untuk mengatur warna garis tepi dan secara *default* bernilai hitam ("black"). Sedangkan untuk `warna isi` digunakan untuk mengatur warna isi (*fill*) dari kotak. Untuk menghilangkan garis tepi maupun isi anda dapat mengatur nilainya menjadi "none", sedangkan untuk gaya atau tipe garis tepi menggunakan opsi "dash".

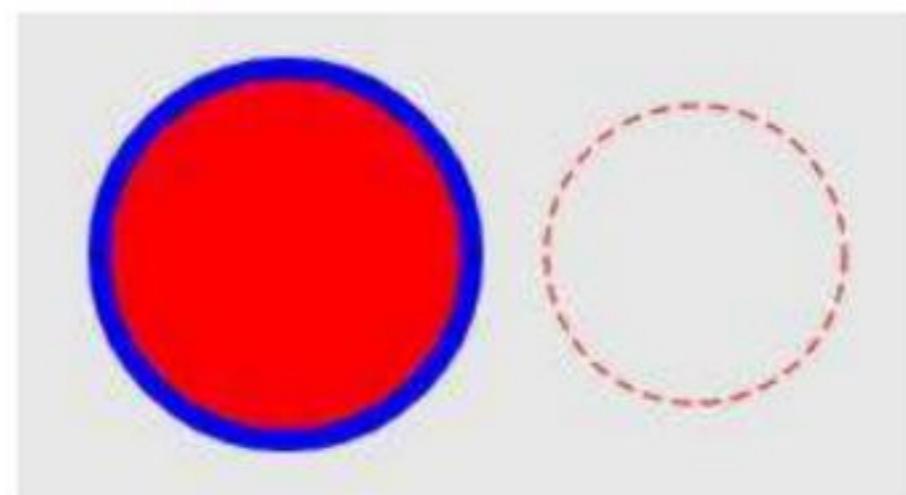
C. Membuat Lingkaran

Seperti halnya fungsi membuat kotak, fungsi membuat lingkaran memiliki struktur sebagai berikut :

```
lingkaran (x, y, r, tebal, warna garis, isi, tipe);
```

Pada fungsi `lingkaran` di atas `x, y` adalah titik pusat lingkaran dan `r` adalah jari-jari dalam satuan *pixel*. Parameter lainnya seperti `tebal` garis, `warna garis` sampai `tipe` memiliki penjelasan yang sama dengan fungsi `kotak`. Sebagai contoh penambahan kode berikut pada file **simulasi-2.js** akan menghasilkan lingkaran sesuai dengan gambar 4.7.

```
16. //membuat lingkaran  
17. lingkaran(150, 250, 50, 6, "blue", "red")  
18. lingkaran(260, 250, 40, 1, "red", "none", "dash")  
19. }
```



Gambar 4.7 Hasil fungsi lingkaran

D. Membuat Diagram Kartesius (Sistem Koordinat Kartesius)

Sistem koordinat kartesius adalah sebuah sistem yang menunjukkan titik titik dalam sebuah bidang dengan menggunakan beberapa bilangan pada garis koordinat. Dalam bidang tersebut terdapat satu set garis referensi, yang memiliki jarak dengan satuan panjang yang sama dan saling tegak lurus.

Sistem koordinat kartesius dalam simulasi digunakan untuk mengetahui hubungan antara dua variabel atau lebih. Sebagai contoh hubungan antara waktu dan jarak pada simulasi kecepatan, menampilkan kurva/garis, persamaan matematika dan sebagainya.

a. **File simulasi-2.js**

Untuk membuat diagram kartesius, perhatikan langkah berikut :

1. Buka kembali **file simulasi-2.js** pada teks editor **Notepad++**.
2. Pada baris 3, di bawah kode :

```
setJudul("Grafis dasar HTML Canvas");
```

Buatlah sebuah variabel baru dengan nama **graf** yang berisi data untuk membuat grafik koordinat kartesius. Perhatikan kode baris 3 di bawah ini

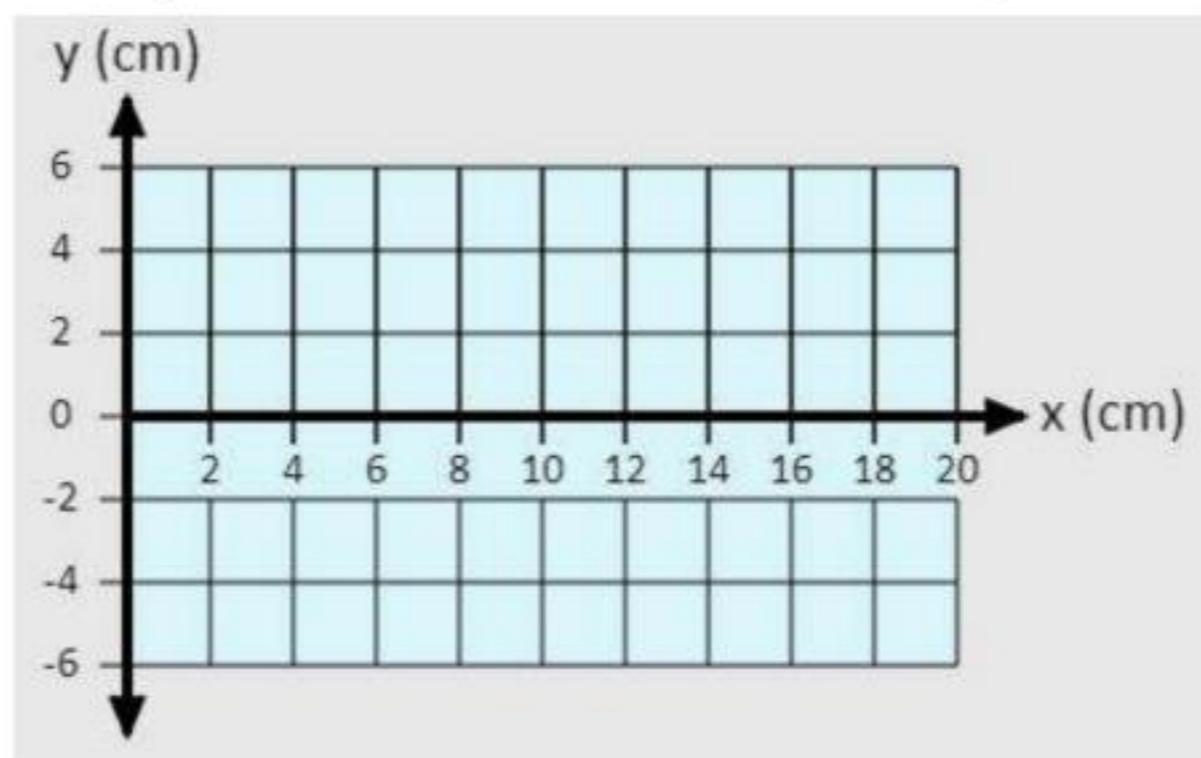
```
2. setJudul("Grafis dasar HTML Canvas");
3. var graf = {startX:80, startY:360, dataW:10, dataH:6,
tileW:30, skalaX:2, skalaY:2, desimalX:0, desimalY:0,
offsetX:0, offsetY:3, xLabel:'x (cm)', yLabel:'y (cm)',
fontLabel:'12pt Calibri', warnaBG:'#daf6fb',
warnaGaris:'#000', warnaLabel:'#000'}
```

3. Selanjutnya pada fungsi **setAwal**, pada baris terakhir sebelum karakter }, tambahkan kode berikut :

```
16. ...
17. //membuat koordinat kartesius
18. grafik(graf);
19. }
```

4. Simpan file.

Jalankan kembali file **tutorial-2.html** dengan membukanya pada *internet browser*, atau tekan tombol *refresh* (F5) apabila file sudah terbuka di *internet browser*. Hasil dari penambahan kode di atas adalah sebagai berikut :



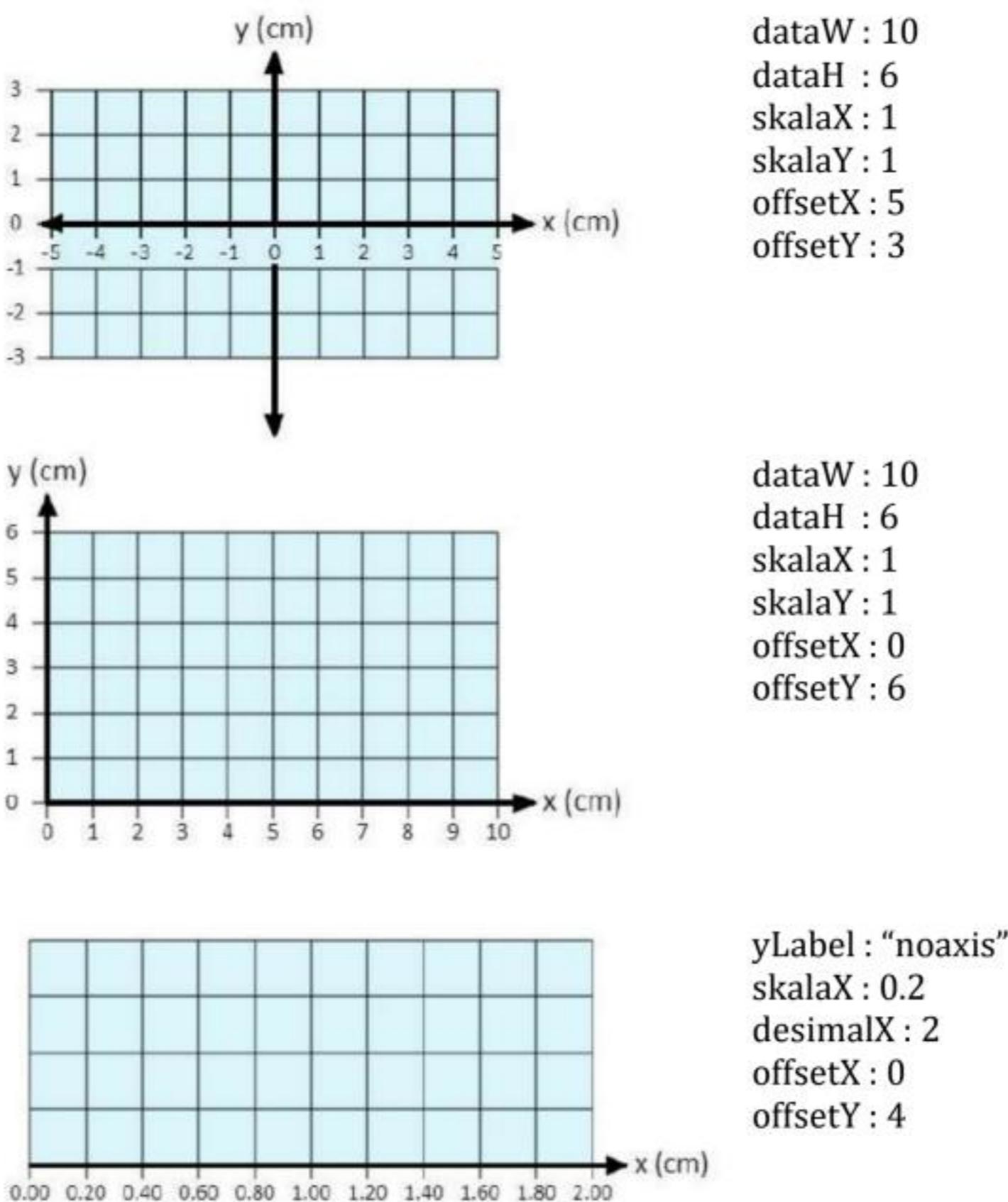
Gambar 4.8 Hasil dari fungsi grafik

b. Penjelasan Program

Untuk membuat diagram kartesian digunakan fungsi *grafik* dengan variabel bertipe objek dengan nama **graf** (baris 3). Adapun variabel tersebut memiliki beberapa pengaturan data, yaitu :

startX	Koordinat x awal dimulainya diagram kartesius
startY	Koordinat y awal dimulainya diagram kartesius
dataW	Jumlah data pada koordinat x (absis)
dataH	Jumlah data pada koordinat y (ordinat)
tileW	Besarnya tiap satuan data dalam <i>pixel</i>
skalaX	Besaran kelipatan nilai data pada sumbu x
skalaY	Besaran kelipatan nilai data pada sumbu y
desimalX	Nilai desimal dari data pada sumbu x
desimalY	Nilai desimal dari data pada sumbu y
offsetX	Pergeseran sumbu x
offsetY	Pergeseran sumbu y
xLabel	Label pada sumbu x
yLabel	Label pada sumbu y
fontLabel	Jenis huruf dan ukurannya untuk menuliskan label
warnaBG	Warna dasar untuk latar belakang diagram
warnaGaris	Warna untuk garis sumbu
warnaLabel	Warna huruf label

Untuk membuat variasi grafis pada diagram kartesian, anda dapat melakukan perubahan data yang ada di dalam variabel tersebut. Sebagai contoh perhatikan gambar grafik dan pengaturannya sebagai berikut :



Gambar 4.9 Fungsi Grafik dan Pengaturannya

E. Menggabungkan Kode *Library* dengan Kode JS

Pada beberapa kasus tertentu, menampilkan grafis membutuhkan kode asli Javascript (*native*). Hal ini mungkin dikarenakan terdapat beberapa perintah yang belum diadaptasi ke dalam *library*, atau perintah yang sudah ada siap pakai dan sudah dalam format yang paling sederhana. Dalam contoh pada tutorial ini misalnya, kita akan menggunakan kode dasar `moveTo`, dan `lineTo` untuk membuat kurva gelombang sinus.

a. File simulasi-2.js

Untuk membuat kurva sinus pada grafik yang sudah kita buat sebelumnya, perhatikan langkah berikut :

1. Buka kembali file **simulasi-2.js** pada teks *editor Notepad++*.
2. pada fungsi **setAwal**, pada baris terakhir sebelum karakter **}**, tambahkan kode berikut :

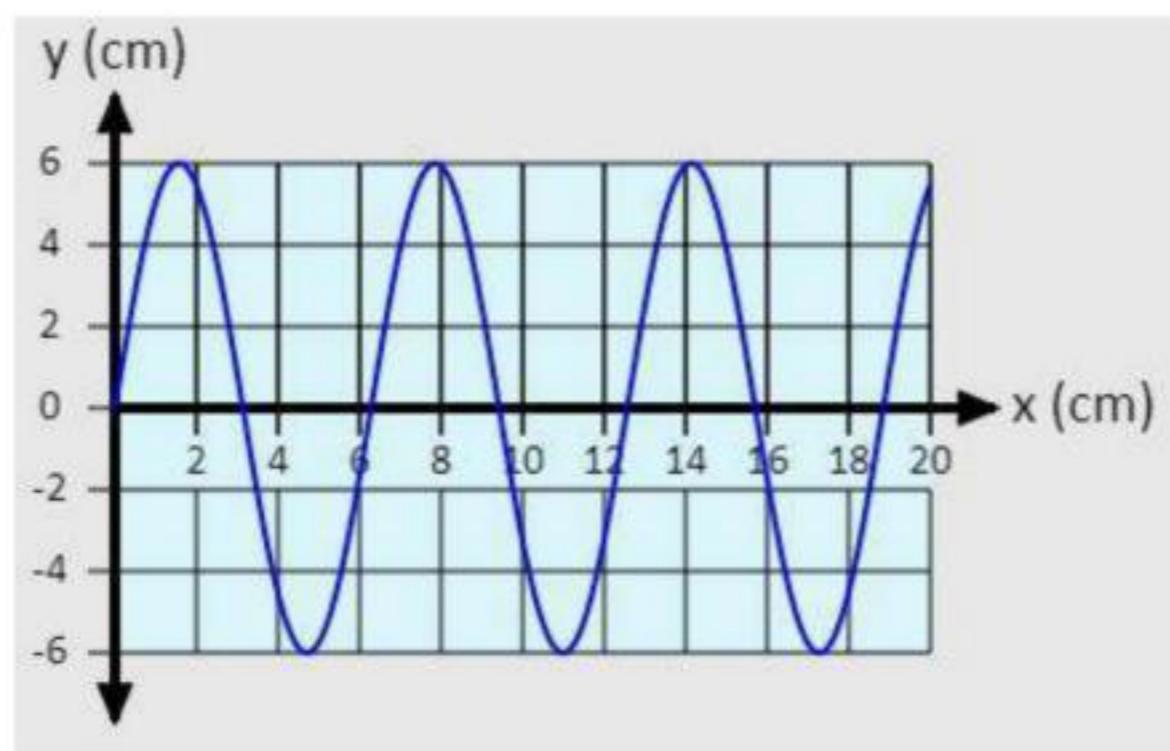
```

18. ...
19. //menggambar garis sinusoidal pada diagram
20. konten.strokeStyle = 'blue';
21. konten.beginPath();
22. konten.moveTo(graf.startX, graf.startY + graf.tileW*
graf.dataH/2);
23. var maxIndex = 1200;
24. for (var i = 0; i <=maxIndex; i++) {
25.     konten.lineTo(graf.startX+i/4, graf.startY+
graf.tileW*graf.dataH/2 - 90*Math.sin(i/60));
26. }
27. konten.stroke();
28. }

```

3. Simpan file.

Jalankan kembali file **tutorial-2.html** dengan membukanya pada *internet browser*, atau tekan tombol *refresh* (F5) apabila file sudah terbuka di *internet browser*. Hasil dari penambahan kode di atas adalah sebagai berikut :



Gambar 4.10 Hasil tutorial menggambar kurva sinus

b. Penjelasan Program

Pada kode di atas, ditambahkan beberapa baris kode *native* dari Javascript yaitu:

- `konten.strokeStyle` yang digunakan untuk mengatur warna garis/kurva
- `konten.beginPath` yang digunakan untuk memulai proses menggambar garis/kurva
- `konten.moveTo` yang digunakan untuk memindah posisi awal menggambar secara spesifik ke koordinat tertentu
- `for loops` yang digunakan untuk mengeksekusi perintah secara berulang-ulang, yang dalam hal ini perintah dilakukan sebanyak 1200 kali (sesuai dengan variabel `maxIndex`).

- `konten.lineTo` yang digunakan untuk membuat *path* garis/kurva sesuai dengan koordinat x, y yang didefinisikan. Dalam hal ini kurva sinus terbentuk akibat perhitungan operasi matematika `Math.sin()` pada baris 25.

```
25. konten.lineTo(graf.startX+i/4, graf.startY+
graf.tileW*graf.dataH/2 - 90*Math.sin(i/60));
```

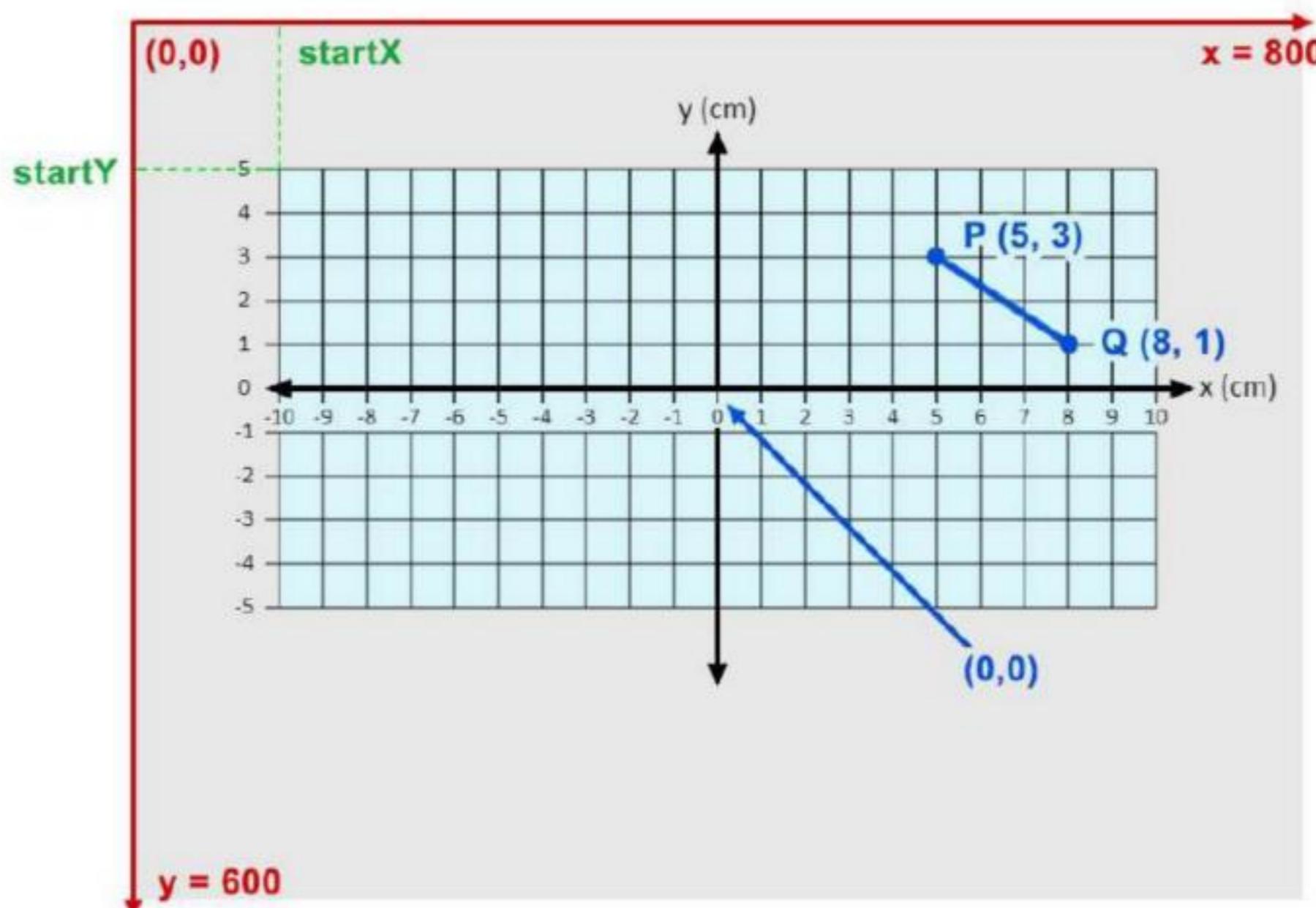
- Pada baris 25 di atas kode, `graf.startX+i/4` angka 4 diperoleh dari variabel `maxIndex` (bernilai 1200) dibagi dengan lebar koordinat kartesian (`graf.tileW*graf.dataW = 30 x 10 = 300`), sehingga kurva sinus yang dibentuk nantinya akan memiliki lebar yang sama dengan lebar koordinat kartesian.
- Selanjutnya di baris 25 di atas, kode `graf.tileW*graf.dataH/2` angka 2 diperoleh dari setengah tinggi grafik kartesian (`graf.tileW*graf.dataH`), sehingga kurva sinus yang dibentuk nantinya akan memiliki tinggi yang sama dengan lebar koordinat kartesian (nilai 2 dikarenakan satu sisi berada di atas garis axis/positif dan satu sisi lainnya berada di bawah garis axis/negatif).
- Selanjutnya tetap di baris 25, kode `90*Math.sin(i/60)` angka 90 diperoleh dari 3 satuan yang digunakan pada grafik kartesian dan setiap satuan bernilai 30 *pixel* (`graf.tileW*graf.dataH/2 = 30 x 6 / 2 = 90`), sehingga kurva sinus yang dibentuk nantinya akan memiliki tinggi 90 *pixel*. Sedangkan angka 60 digunakan untuk memperbesar interval gelombang sinus (anda dapat mengubah nilai tersebut untuk mengetahui perbedaan kurva yang dihasilkan). Penjelasan tentang transformasi sistem koordinat kartesian ke dalam sistem koordinat layar, terdapat pada sub bab berikutnya.
- `konten.stroke()` yang digunakan untuk menampilkan garis yang telah disusun melalui *path* yang telah diatur pada seluruh kode setelah `beginPath` dieksekusi.

Yang dimaksud dengan `konten` pada kode di atas merupakan konteks dari HTML canvas. Pada tutorial berbahasa Inggris, biasanya variabel tersebut ditulis dengan nama `context` atau `ctx`.

4.4 Transformasi Sistem Koordinat Layar

Dalam contoh tutorial 2 di atas, terdapat satu konsep yang harus dipahami oleh pengembang aplikasi bahwa sistem koordinat layar berbeda dengan sistem koordinat kartesian. Sistem koordinat layar merupakan sistem koordinat yang menggunakan satuan *pixel* (titik). Ketika mendefinisikan sebuah canvas, kita juga mendefinisikan ukuran dari canvas tersebut, dan sistem koordinat layar mengikuti ukuran canvas dengan titik pusat $(0,0)$ berada pada ujung kiri atas layar (garis warna merah pada gambar 4.11). Pada koordinat layar x bernilai positif ke arah kanan, sedangkan koordinat layar y bernilai positif jika ke arah bawah.

Sementara itu koordinat kartesian, yang secara umum digunakan pada bidang sains memiliki sistem yang berbeda dimana koordinat y (ordinat) akan bernilai positif jika ke arah atas, dan bernilai negatif jika ke arah bawah. Hal inilah yang menyebabkan kita harus mengatur transformasi sistem koordinat kartesian ke dalam sistem koordinat layar. Penggambaran koordinat kartesian ke dalam sistem koordinat layar harus dipetakan (ditransformasikan) dalam bentuk plot *pixel*. Perhatikan gambar berikut :



Gambar 4.11 Transformasi sistem koordinat

Pada gambar di atas, kita ingin menarik garis PQ , dimana garis PQ terdapat pada sistem koordinat kartesian. Sementara itu, koordinat kartesian tersebut dibentuk

oleh fungsi grafik yang berada di dalam *library* simulasidasar.js dalam satuan *pixel*. Maka koordinat titik P dan koordinat titik Q perlu dihitung terlebih dahulu menyesuaikan atribut yang dimiliki oleh sistem koordinat kartesian. Sistem koordinat kartesian di atas dibentuk oleh kode berikut :

```
1. var graf = {startX:100, startY:100, dataW:20, dataH:10,  
tileW:30, skalaX:1, skalaY:1, desimalX:0, desimalY:0,  
offsetX:10, offsetY:5, xLabel:'x (cm)', yLabel:'y  
(cm)', fontLabel:'12pt Calibri', warnaBG:'#daf6fb',  
warnaGaris:'#000', warnaLabel:'#000'}
```

Melalui variabel di atas diketahui bahwa setiap satuan skala pada koordinat kartesian dibentuk atas 30 satuan *pixel* (*tileW*). Kemudian pergeseran sumbu axis (x) adalah 10 satuan skala (*offsetX*) dan pergeseran sumbu ordinat (y) adalah 5 satuan skala (*offsetY*). Dengan demikian untuk mendapatkan pusat dari koordinat kartesian, kita dapat menggunakan rumus transformasi linear, yaitu dengan formula :

```
x0 = startX + (offsetX*tileW);  
y0 = startY + (offsetY*tileW);
```

Setelah titik pusat koordinat kartesian (0, 0) ditemukan, maka dengan mudah kita dapat menemukan titik P dan Q melalui perhitungan transformasi linear sebagai berikut ;

```
Px1 = x0+(Px*tileW); Py1 = y0-(Py*tileW);  
Qx1 = x0+(Qx*tileW); Qy1 = y0-(Qy*tileW);
```

Perhatikan bahwa kordinat Y pada koordinat layar berbanding terbalik dengan koordinat Y pada koordinat kartesian, sehingga perhitungan koordinat Y menggunakan operasi pengurangan.

Apabila kita melakukan penskalaan pada koordinat kartesian tersebut, dengan mengubah nilai *skalaX* dan *skalaY*, maka transformasi linear yang dilakukan harus melibatkan variabel tersebut menjadi sebagai berikut :

```
Px1 = x0+(Px*tileW/skalaX); Py1 = y0-(Py*tileW/skalaY);  
Qx1 = x0+(Qx*tileW/skalaX); Qy1 = y0-(Qy*tileW/skalaY);
```

Dengan ditemukannya titik P dan titik Q pada sistem koordinat layar, maka dengan mudah kita dapat membuat garis PQ dengan fungsi garis sebagai berikut :

```
garis (Px1, Py1, Qx1, Qy1);
```

Untuk lebih memudahkan anda dalam memahami transformasi koordinat kartesian ke koordinat layar, cobalah ikuti tutorial berikut :

A. Tutorial 2c - Membuat Garis pada Sistem Koordinat Kartesian

a. *File HTML*

1. Copy file **tutorial-2.html** dan *paste* ke dalam *folder* tutorial 2.
2. *Rename* nama *file* menjadi **tutorial-2c.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

12. <script src="**simulasi-2c.js**"></script>

3. Simpan kembali *file* HTML.

b. *File simulasi-2c.js*

1. Pada Notepad++, buatlah sebuah *file* baru.

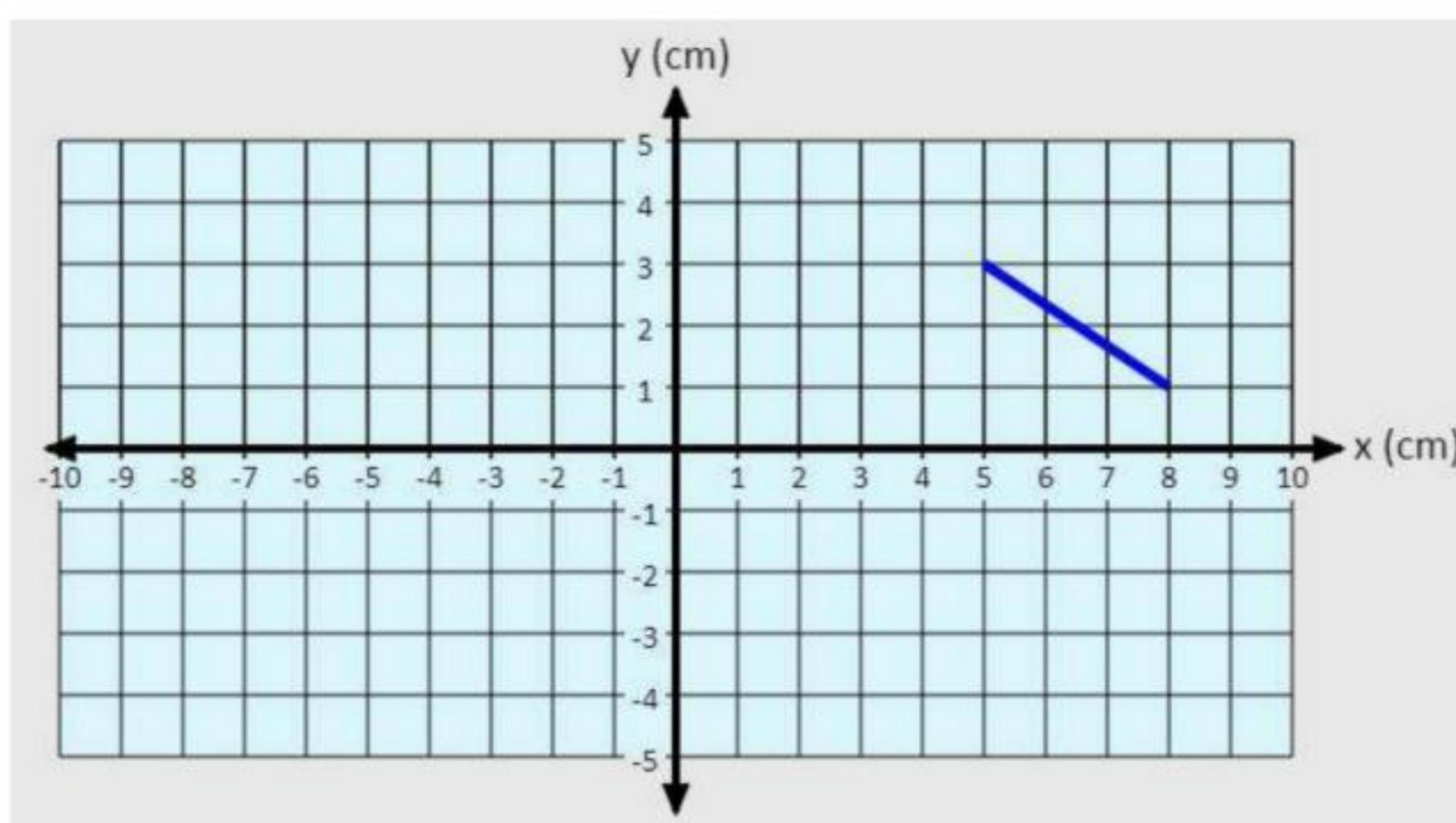
2. Ketikan kode berikut :

```
1. aturCanvas();
2. setJudul("Membuat garis pada koordinat kartesian");
3.
4. var graf = {startX:100, startY:100, dataW:20, dataH:10,
   tileW:30, skalaX:1, skalaY:1, desimalX:0, desimalY:0,
   offsetX:10, offsetY:5, xLabel:'x (cm)', yLabel:'y
   (cm)', fontLabel:'12pt Calibri', warnaBG:'#daf6fb',
   warnaGaris:'#000', warnaLabel:'#000'}
5.
6. var P = {x:5, y:3};
7. var Q = {x:8, y:1};
8.
9. function setSimulasi(){
10.    //menambahkan background warna
11.    hapusLayar("#e8e8e8");
12.    //membuat kordinat kartesius
13.    grafik(graf);
14.    //titik pusat koordinat kartesian
15.    var x0 = graf.startX + graf.offsetX*graf.tileW;
16.    var y0 = graf.startY + graf.offsetY*graf.tileW;
17.    //transformasi titik P dan Q ke koordinat layar
18.    var Px1 = x0 + (P.x*graf.tileW/graf.skalaX);
19.    var Py1 = y0 - (P.y*graf.tileW/graf.skalaY);
20.    var Qx1 = x0 + (Q.x*graf.tileW/graf.skalaX);
21.    var Qy1 = y0 - (Q.y*graf.tileW/graf.skalaY);
22.    //membuat garis
23.    garis (Px1, Py1, Qx1, Qy1, 5, "blue");
24. }
25.
26. setSimulasi();
```

3. Simpan dengan nama **simulasi-2c.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-2.

Jalankan **file tutorial-2c.html**, dengan cara membukanya di internet *browser*.

Maka kode di atas akan menampilkan hasil sebagai berikut :



Gambar 4.12 Hasil tutorial 2c.

Dengan memahami prinsip koordinat layar dan koordinat kartesian yang dihasilkan oleh fungsi *grafik*, serta transformasi linear yang harus dilakukan, maka kita dapat menggunakan koordinat kartesian tersebut dalam simulasi yang lebih kompleks.

Pada bab berikutnya akan dijelaskan mengenai fungsi pada *library* *simulasDasar.js* untuk menyusun grafis yang lebih kompleks, seperti fungsi untuk membuat roda gigi, gelas kimia, termometer, partikel api, partikel gas, partikel air, dan sebagainya. Fungsi-fungsi ini nantinya akan dijelaskan seiring dengan simulasi yang akan dibuat melalui tutorial-tutorial yang ada di dalam buku ini.

BAB 5

Operasi Teks dengan Javascript

5.1 Fungsi *fillText* pada canvas

Teks merupakan elemen antarmuka (*interface*) yang penting dalam sebuah aplikasi, dimana teks memberikan informasi tertulis kepada pengguna. Dalam HTML canvas, fitur teks *native* relatif terbatas jika dibandingkan dengan bahasa pemrograman yang lain. Adapun fungsi untuk menampilkan teks pada Javascript adalah sebagai berikut :

```
konten.font = "12pt Calibri";
konten.fillStyle = "black";
konten.textAlign = "left";
konten.fillText("Helloworld !", 100, 100);
```

Fungsi *fillText* tersebut akan menampilkan teks pada canvas. Pengaturan dalam fungsi ini termasuk sangat sederhana, mengingat sebuah aplikasi (terlebih aplikasi interaktif), pada umumnya memiliki pengaturan yang lebih kompleks, seperti pengaturan *text area*, pengaturan paragraf, *input text*, teks bertipe HTML dan sebagainya. Oleh karena itu pengembangan fitur operasi teks pada umumnya menggunakan *library* populer seperti JQuery, paperjs, react dan sebagainya. Dalam kode *library* simulasiDasar.js juga dikembangkan fungsi teks agar lebih fleksibel untuk digunakan di dalam simulasi.

5.2 Fungsi teks pada simulasiDasar.js

Tutorial ketiga pada buku ini akan menjelaskan fungsi dasar dalam mengelola teks pada kanvas. Perhatikan langkah berikut:

A. Tutorial 3a - Operasi Teks

a. File HTML

1. Seperti pada projek tutorial-2, buatlah sebuah *folder* khusus dengan nama **tutorial-3**. Copy file **tutorial-2.html** dan *paste* ke dalam *folder*.
2. *Rename* nama *file* menjadi **tutorial-3.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
12.    <script src="simulasi-3.js"></script>
```

3. Simpan kembali *file* HTML.

b. File library

1. *Copy paste file* **simulasiDasar.js** ke dalam *folder tutorial-3*.

c. File simulasi-3.js

1. Pada Notepad++, buatlah sebuah *file* baru.
2. Ketikan kode berikut :

```
1. aturCanvas();  
2. setJudul("Teks pada HTML Canvas");  
3.  
4. function setSimulasi(){  
5. //menambahkan background warna  
6. hapusLayar("#e8e8e8");  
7. //menampilkan teks  
8. teks("Operasi Teks", 0.5*(canvas.width), 40);  
9. teks("Fungsi untuk mengatur tampilan teks pada  
    canvas", 250, 60, "12pt Calibri", "#bf0433", "left");  
10. }  
11. setSimulasi();
```

3. Simpan dengan nama **simulasi-3.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder tutorial-3*.

Jalankan *file* **tutorial-3.html**, dengan cara membukanya di *internet browser*. Maka kode di atas akan menampilkan teks sebagai berikut :



Gambar 5.1 Hasil penambahan teks pada canvas

d. Penjelasan Program

Fungsi *teks* merupakan fungsi standar untuk menyederhanakan kode *fillText*, dimana fungsi ini digunakan untuk menampilkan 1 baris teks. Adapun struktur fungsi *teks* adalah sebagai berikut :

```
teks(txt, x, y, huruf, warna, perataan teks);
```

Pada baris 8, parameter yang dimasukkan ke dalam fungsi *teks* hanya *txt* ("operasi teks"), dan kordinat munculnya teks yaitu *x* setengah dari *canvas* (tepat di tengah kanvas akibat kode $0.5 \times \text{canvas.width}$), serta *y* = 40. Secara *default* kode *teks* menggunakan huruf *Calibri 14 pt bold* berwarna hitam.

Pada baris 9 pengaturan lebih kompleks dilakukan dengan menambahkan jenis huruf, warna dan perataan teks kiri (*text alignment*). Untuk pengaturan huruf, pada fungsi `teks` hanya bisa dilakukan 1 jenis saja, sehingga jika anda ingin menuliskan kata yang memiliki 2 atau lebih jenis huruf tidak dapat dilakukan dengan menggunakan fungsi `teks`. Misal anda ingin menulis :

Percobaan **fisika** membutuhkan keterampilan khusus.

Pada teks di atas, terdapat 2 jenis huruf, yaitu huruf normal (*reguler*) dan huruf tebal (*bold*), sehingga tidak bisa dilakukan. Fitur ini bisa dilakukan dengan fungsi lain, yaitu `teksHTML` yang akan dijelaskan pada tutorial selanjutnya.

Dalam fungsi `teks`, untuk mengatur huruf digunakan format sebagai berikut :

- “bold 11pt Calibri” akan menghasilkan huruf **Calibri tebal berukuran 11 point**.
- “italic 12pt Cambria” akan menghasilkan huruf *Cambria miring berukuran 12 point*.
- “11pt Arial italic” akan menghasilkan huruf serif (Times New Roman), karena kata *italic/bold* harus berada di depan, dan ketika huruf tidak terdefinisikan dengan tepat, maka aplikasi akan menampilkan huruf dasar yaitu Times New Roman.

Fitur `underline` (teks bergaris bawah) tidak didukung oleh canvas, sehingga harus dilakukan secara manual dengan menarik garis menggunakan fungsi `garis`.

B. Tutorial 3b - `teksHTML`

Untuk membuat *file* dengan pengaturan yang lebih kompleks, seperti pada kasus penggunaan huruf tebal, miring, dan reguler secara bersamaan, serta untuk membuat *file* dengan pengaturan paragraf, maka digunakan fungsi `teksHTML`. Perhatikan contoh berikut :

a. **File simulasi-3.js**

1. Lanjutkan mengedit **file simulasi-3.js**.
2. Pada baris 3, di bawah kode `setJudul` sisipkan kode berikut :
 1. `aturCanvas(setSimulasi);`
 2. `setJudul("Teks pada HTML Canvas");`
 3. `var paragraf = "Dalam Sistem Internasional,<i>F</i> diukur dalam <i>newton</i> (N),<i>m₁</i> dan <i>m₂</i> dalam`

kilogram (kg), r dalam meter (m), dan konstanta G kira-kira sama dengan $6,67 \times 10^{-11} \text{ N m}^2 \text{ kg}^{-2}$.";

3. Pada fungsi `setSimulasi()`, sebelum karakter terakhir } , sisipkan kode berikut :

```
5. function setSimulasi(){  
6.     ...  
11.     teksHTML(paragraf, 100, 120, 400,  
12.         "14pt-Cambria-left-1.6", "#000");  
12. }
```

4. Simpan file dan refresh internet browser, maka akan didapatkan hasil sebagai berikut :

Dalam **Sistem Internasional**, F diukur dalam newton (N), m_1 dan m_2 dalam kilogram (kg), r dalam meter (m), dan konstanta G kira-kira sama dengan $6,67 \times 10^{-11} \text{ N m}^2 \text{ kg}^{-2}$.

Gambar 5.2 Hasil fungsi `teksHTML`

b. Penjelasan Program

Fungsi `teksHTML` menghasilkan luaran sebagaimana fungsi `teks` namun dapat mengubah tag html standar untuk teks seperti *bold* (``), *italic* (`<i>`), *subscript* (`<sub>`), *superscript* (`<sup>`) dan enter (`
`). Dengan cara ini kita dapat menuliskan kalimat dengan pengaturan yang lebih kompleks.

Pengaturan huruf pada fungsi `teksHTML` juga sedikit berbeda dengan menggunakan karakter “-” sebagai pemisah. Sebagai contoh pada pengaturan “14pt-Cambria-left-1.6” di atas akan menghasilkan huruf Cambria berukuran 14 *point*, perataan kiri dan spasi antar baris sebesar 1,6 kali dari tinggi huruf.

C. Tutorial 3b – Membuat Input Teks

Dalam beberapa aplikasi, sering kali kita dapatkan aplikasi meminta pengguna untuk memasukkan data pengguna seperti nama, umur, alamat dalam format formulir (*form*). Data tersebut dimasukkan ke dalam sebuah area yang disebut dengan input teks. Untuk membuat input teks, perhatikan langkah-langkah berikut:

a. File simulasi-3.js

1. Lanjutkan mengedit **file simulasi-3.js**.
2. Pada baris 3, di bawah kode var paragraf = .. sisipkan kode berikut :

```
4. canvas.onmouseup = mouseUp;
5. //data teks input
6. var input1 = {nama:"nama", x:100, y:250, p:120, t:30,
   huruf:"13pt Calibri", val:"nama", max:30, limit:"*"};
7. var input2 = {nama:"umur", x:100, y:320, p:120, t:30,
   huruf:"13pt Calibri", val:"15", max:2, limit:"0-9"};
8. //data popup
9. var popup1 = {x:200, y:200, l:400, t:200,
   warnaBG:"#f7f7f7", warnaGaris:"#bababa", val:"",
   huruf:"14pt-Calibri-center-1.5", warnaHuruf:"#8c1515",
   tutup:"ok", func: setSimulasi}
```

3. Pada fungsi setSimulasi(), sebelum karakter terakhir } , sisipkan kode berikut :

```
13. function setSimulasi(){
14. ...
21.  teks("Nama", 100, 240, "bold 14pt Calibri", "black",
   "left");
22.  teks("Umur", 100, 310, "bold 14pt Calibri", "black",
   "left");
23.  teksInput(input1);
24.  teksInput(input2);
25.  tombol("OK", 130, 360, 50, 30, "bold 14pt Calibri",
   "white", "black", "gray", "r");
26. }
```

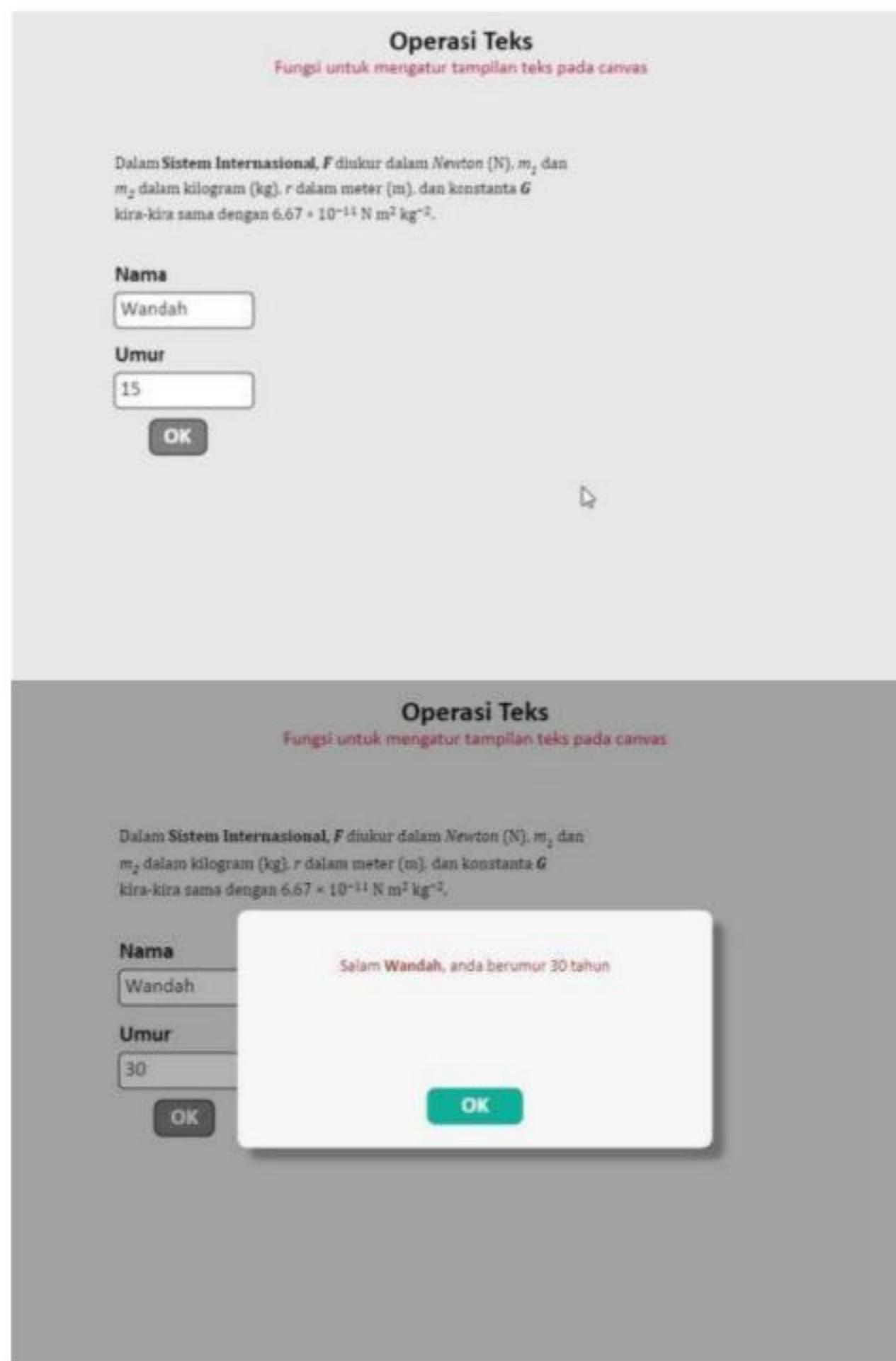
4. Di bawah, akhir dari fungsi setSimulasi, tambahkan fungsi baru untuk mendeteksi gerakan mouse, sebagai berikut :

```
28. function mouseUp(event){
29.   //prosedur mengecek input text dan popup
30.   cekPopup(event);
31.   cekTeksInput(event);
32.   var tombolAktif = cekTombol(event);
33.   if (tombolAktif == "OK"){
34.     if (input1.val == "nama"){
35.       popup1.val = "Anda belum mengetikan nama,
         silahkan <b>klik pada kolom ama<b> dan ketikan
         nama anda!";
36.     }else{
37.       popup1.val = "Salam <b>"+input1.val+"</b>,
         anda berumur "+input2.val+" tahun";
38.     }
39.     setSimulasi();
40.     popup(popup1);
41.   }
42. }
```

43.

```
44. setSimulasi();
```

5. Simpan file dan refresh internet browser, maka akan didapatkan hasil sebagai berikut :



Gambar 5.3 Input teks dan *popup*

b. Penjelasan Program

Fungsi `teksInput` diawali dengan mendefinisikan data yang dibutuhkan melalui sebuah variabel, dimana dalam tutorial ini terdapat 2 variabel, yaitu variabel `input1` dan `input2` (baris 6 dan 7). Adapun parameter yang terdapat pada variabel untuk membentuk input teks adalah sebagai berikut :

nama	Digunakan untuk mengidentifikasi nama dari input teks. Masing-masing input teks harus memiliki nama yang unik agar tidak terjadi kesalahan pengecekan ketika fungsi cekTeksInput dijalankan
x	Kordinat x dari input teks
y	Kordinat y dari input teks
p	Panjang area input teks dalam satuan <i>pixel</i>
t	Tinggi area input teks dalam satuan <i>pixel</i>
huruf	Jenis huruf yang digunakan untuk input teks. Format huruf mengikuti fungsi teks contoh “bold 13pt Calibri”.
val	Nilai dari input teks. Pada contoh di atas variabel <code>input1</code> bernilai “nama”, maka ketika aplikasi dijalankan pada input teks telah terisi dengan tulisan “nama”. Anda dapat menuliskan teks kosong (“ ”) untuk mengawali teks input tanpa tulisan apapun.
Selanjutnya nilai <code>val</code> dideteksi pada baris 34 dengan kode <code>input1.val == " "</code> . Nilai <code>val</code> akan terupdate secara otomatis setelah diinputkan oleh pengguna.	
max	Merupakan jumlah karakter maksimal yang dapat diinputkan oleh pengguna. Pada variabel <code>input2</code> misalnya parameter <code>max</code> bernilai 2, sehingga pengguna hanya dapat menginputkan maksimal 2 karakter
limit	Merupakan batasan karakter yang dapat diinputkan oleh pengguna. Pada variabel <code>input1</code> misalnya parameter <code>limit</code> bernilai *, artinya tidak ada batasan karakter, sehingga anda dapat mengetikan huruf dan angka.
<p>Sedangkan pada variabel <code>input2</code> parameter <code>limit</code> bernilai 0-9, artinya hanya bisa diinput dengan angka 0-9.</p> <p>Adapun jenis limit yang dapat terdeteksi adalah :</p> <ul style="list-style-type: none"> * tanpa batasan karakter A-Z atau a-z hanya bisa diinput huruf 0-9 hanya bisa diinput angka 	

Selanjutnya untuk menampilkan input teks ke canvas, digunakan fungsi `teksInput`, sebagaimana pada baris 23 dan 24

```
23.  teksInput(input1);
24.  teksInput(input2);
```

Fungsi selanjutnya yang diperkenalkan dalam tutorial di atas adalah fungsi `popup`, yang digunakan untuk menampilkan kotak peringatan atau yang diistilahkan dengan *popup*. Seperti halnya teks input, untuk mengatur `popup` juga diperlukan sebuah variabel `popup1` dengan pengaturan parameter seperti pada baris 9.

x	Kordinat x dari munculnya <i>popup</i>
y	Kordinat y dari munculnya <i>popup</i>
l	Lebar <i>popup</i> dalam satuan <i>pixel</i>
t	Tinggi <i>popup</i> dalam satuan <i>pixel</i>
huruf	Jenis huruf yang digunakan untuk <i>popup</i> . Format huruf mengikuti fungsi <code>teksHTML</code> contoh “14pt-Calibri-center-1.5”. Berbeda dengan <code>teksInput</code> , <i>popup</i> mendukung format HTML
warnaHuruf	Warna huruf yang digunakan dalam <i>popup</i>
warnaBG	Warna latar belakang area <i>popup</i>
warnaGaris	Warna garis tepi dari <i>popup</i>
val	Nilai dari <i>popup</i> . Pada contoh di atas <i>value</i> dari <i>popup</i> diatur pada baris 35 dan 37, dimana digunakan teknik <code>teksHTML</code> dengan menambahkan tag <code></code> untuk menebalkan huruf.
	Pada baris tersebut kode mengecek nilai dari variabel <code>input1</code> , dan jika pengguna telah mengisi variabel <code>input1</code> dengan benar maka <i>popup</i> akan menampilkan value pada baris ke 35 dan jika value dari <code>input1</code> masih kosong atau masih berisi dengan “ <i>nama</i> ”, maka <i>value</i> di set pada baris ke 36.
tutup	Merupakan label dari tombol <i>popup</i> . Tombol yang dimaksud adalah tombol yang digunakan untuk menutup <i>popup</i>
func	Merupakan fungsi yang akan dijalankan setelah <i>popup</i> tertutup. Dalam hal ini, setelah <i>popup</i> tertutup, fungsi <code>setSimulasi</code> akan dijalankan kembali, sehingga pengguna dapat mengedit isian <code>input1</code> dan <code>input2</code> kembali.

Untuk menampilkan *popup* tersebut ke canvas, digunakan kode `popup(data popup);`, yang dapat dilihat pada baris 40.

```
40.      popup(popup1);
```

Fungsi `teksInput` dan `tombol` untuk submit form pada tutorial di atas, memerlukan elemen interaktivitas yaitu masukan dari pengguna. Dalam hal ini masukan yang dimaksud adalah input yang berasal dari *mouse* (klik *mouse*). Oleh karena itu, dibutuhkan sebuah kode untuk mendeteksi *mouse*, yaitu ketika *mouse* ditekan. Kode ini diletakkan pada awal baris, yaitu pada baris 4 :

```
4.  canvas.onmouseup = mouseUp;
```

Pada baris tersebut digunakan *event* `onmouseup`, yaitu mendeteksi ketika tombol kiri *mouse* ditekan dan dilepaskan (klik). Kode di atas, akan menjalankan fungsi `mouseUp` setiap kali mouse di klik. Fungsi `mouseUp` selanjutnya didefinisikan pada baris 28 sampai 42, dimana di dalamnya terdapat 3 perintah pengecekan, yaitu :

```
30.    cekPopup(event);  
31.    cekTeksInput(event);  
32.    var tombolAktif = cekTombol(event);
```

Dalam tutorial di atas, terdapat 3 hal yang harus dicek setiap kali tombol *mouse* ditekan, yaitu teksinput, tombol OK untuk mensubmit formulir, dan tombol OK untuk menutup *popup*. Ketiga elemen interaktif tersebut dicek menggunakan 3 kode di atas. Secara lebih spesifik teknik mengecek elemen interaktif akan dibahas pada bab selanjutnya.

5.3 Formula Matematika dengan *Latex Equation*

Dalam menampilkan informasi atau deskripsi sebuah simulasi, seringkali dilibatkan formula matematika yang melibatkan simbol-simbol khusus. Sebagai contoh untuk menuliskan formula berikut :

$$y = \sum_{x=0}^{10} x^2$$

Menuliskan rumus di atas dengan menggunakan metode `teks / fillText` akan sangat rumit karena membutuhkan karakter dan posisi yang beragam. Oleh karena itu diperlukan format yang lebih mudah, salah satunya dengan memanfaatkan LaTeX. LaTeX adalah bahasa *markup* atau sistem penyiapan dokumen untuk peranti lunak TeX, yaitu program komputer yang digunakan untuk membuat *typesetting* suatu dokumen, atau membuat formula matematika. LaTeX memungkinkan penulis/penggunanya untuk melakukan *typesetting* dan mencetak hasil kerjanya dalam bentuk layout atau tampilan yang memiliki standar tinggi.

Untuk membuat formula dengan LaTeX, diperlukan simbolisasi karakter. Sebagai contoh, rumus di atas dituliskan dalam format LaTeX sebagai berikut :

$$y = \sum_{x=0}^{10} x^2$$

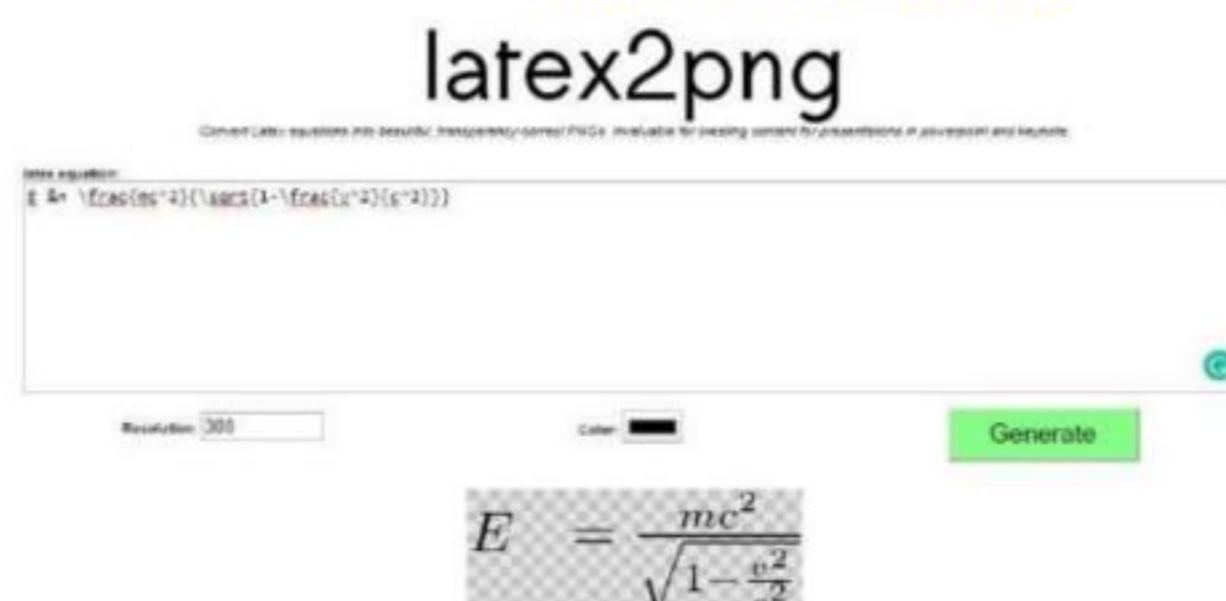
Format tersebut diterjemahkan oleh aplikasi LaTeX, sehingga akan dihasilkan tampilan sebagaimana pada halaman sebelumnya.

Adaptasi LaTeX ke dalam bahasa Javascript dapat ditemukan pada beberapa *library*, beberapa di antaranya adalah mathjax, katex.js, texlive.js, latex.js dan beberapa lainnya. *Library* ini mempermudah menuliskan kode LaTeX dan menampilkannya

dalam format HTML atau SVG, namun sebagian besar (bahkan sejauh saya mencoba beberapa *library* tersebut), semua *library* tersebut membutuhkan waktu beberapa mili detik untuk mengubah kode LaTeX menjadi tampilan rumus yang sempurna. Semakin kompleks rumus yang diinputkan, semakin lama proses mengkonversinya ke format LaTeX. Hal inilah yang menyebabkan saya kurang menyarankan metode konversi secara langsung formula LaTeX ke dalam canvas.

Untuk menampilkan formula matematika kompleks, saya menyarankan untuk mengkonversi formula LaTeX ke format gambar (JPG atau PNG), yang selanjutnya gambar tersebut ditampilkan ke dalam canvas. Metode ini jauh lebih cepat, karena formula dalam bentuk gambar hanya membutuhkan satu kali proses *loading*, sehingga dapat ditampilkan secara *realtime*.

Untuk mengkonversi rumus/formula latex ke dalam format gambar, anda dapat menggunakan *converter online* seperti <http://latex2png.com/>.



Gambar 5.4 *Converter* latex menjadi *bitmap* PNG

Metode berikutnya yang dapat dilakukan, untuk menampilkan formula latex secara cepat ke canvas adalah dengan mengkonversi formula latex ke dalam format SVG dan mengimport file SVG tersebut saat pertama kali aplikasi dijalankan. Metode ini sangat mungkin untuk dilakukan dengan menggunakan beberapa *library* seperti canvas-latex.js yang digabungkan dengan *library* pixy.js atau create.js, yang mana untuk mengatur dan menerapkannya dibutuhkan pengaturan dengan level yang lebih tinggi.

Dalam buku ini digunakan teknik yang paling mudah dan cepat, yaitu menambahkan rumus atau formula yang sudah terformat dalam bentuk JPG atau PNG ke dalam canvas.

BAB 6

Interaktivitas

6.1 Membuat tombol pada canvas

Dalam sebuah media simulasi, pengguna akan berinteraksi dengan sebuah aplikasi melalui tampilan antarmuka (*user interface*). Elemen antarmuka untuk membentuk interaktivitas sangat beragam, mulai dari yang paling sederhana seperti tombol, sampai yang membutuhkan pengaturan lebih kompleks seperti *drag and drop* objek, *slider*, dan interaksi baru seperti sentuhan (*touch*) dan deteksi gerak (*gesture*).

Tutorial keempat pada buku ini akan menjelaskan fungsi dasar untuk membentuk interaktivitas, dimulai dari yang paling sederhana yaitu membuat tombol.

A. Tutorial 4a – Membuat tombol

a. File HTML

1. Seperti pada projek tutorial sebelumnya, buatlah sebuah *folder* khusus dengan nama **tutorial-4**. *Copy file tutorial-3.html* dan *paste* ke dalam *folder*.
2. *Rename* nama *file* menjadi **tutorial-4.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
12. <script src="simulasi-4.js"></script>
```

3. Simpan kembali *file* HTML.

b. File library

1. *Copy paste* *file* **simulasiDasar.js** ke dalam *folder* **tutorial-4**.

c. File simulasi-4.js

1. Pada Notepad++, buatlah sebuah *file* baru
2. Ketikkan kode berikut :

```
1. aturCanvas();  
2. setJudul("Interaktivitas Dasar pada HTML Canvas");  
3. //listener untuk membaca event mouse  
4. canvas.onmouseup = mouseUp;  
5.  
6. var teksLabel = "Klik salah satu tombol";
```

```

7.
8. function setSimulasi(){
9.   hapusLayar("#e8e8e8");
10.  teks("Tombol dan Slider", 0.5*(canvas.width), 50);
11.  teks(teksLabel, 250, 105, "12pt Calibri", "black",
12.    "left");
13.  //membuat tombol
14.  tombol("Tombol 1", 100,80, 120, 40);
15.  tombol("Tombol 2", 100,130, 120, 40, "13pt Arial",
16.    "white", "red", "gray");
17.  tombol("Tombol 3", 100,180, 120, 40, "12pt Calibri",
18.    "white", "black", "gray", "r");
19. }
20. function mouseUp(event){
21.   //prosedure mengecek tombol
22.   var tombolAktif = cekTombol(event);
23.   if (tombolAktif != ""){
24.     teksLabel = "Anda menekan "+tombolAktif;
25.     setSimulasi();
26.   }
27. setSimulasi();

```

3. Simpan dengan nama **simulasi-4.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-4.

Jalankan *file* **tutorial-4.html**, dengan cara membukanya di *internet browser*. Maka kode di atas akan menampilkan 3 variasi tombol sebagai berikut :



Gambar 6.1 Hasil penambahan tombol pada canvas

d. Penjelasan Program

Fungsi `tombol` merupakan fungsi dasar yang digunakan untuk menampilkan sebuah tombol. Adapun struktur fungsi `tombol` adalah sebagai berikut :

```

tombol(label, x, y, l, t, huruf, warna huruf, warna
garis tepi, warna latar, lengkung);

```

Pada baris 13 parameter yang dimasukkan ke dalam fungsi tombol hanya label, kordinat munculnya tombol yaitu x dan y, serta ukuran tombol. Pada pengaturan yang lebih kompleks seperti baris 14 dan 15 anda dapat menambahkan penggayaan huruf dan warna tombol.

label tombol digunakan sebagai identitas masing-masing tombol, yang akan digunakan untuk pengecekan pada baris 20. Apabila dalam satu layar terdapat nama label yang sama, maka dapat ditambahkan karakter "/id=" untuk menambahkan identitas yang unik. Sebagai contoh, terdapat 2 tombol OK di layar, maka kita dapat memberikan label pada tombol sebagai berikut :

- ```
1. tombol("OK/id=ok_1", 100,80, 120, 40);
2. tombol("OK/id=ok_2", 100,80, 120, 80);
```

Maka fungsi cekTombol nantinya akan mendeteksi karakter di belakang "/id=", sehingga label yang diidentifikasi pada contoh di atas adalah "ok\_1" dan "ok\_2".

Opsi lengkung pada fungsi tombol dapat anda isi dengan karakter "r", untuk menghasilkan tombol dengan ujung melengkung (*rounded*).

Parameter terakhir yang dapat ditambahkan pada tombol adalah parameter fungsi, yang akan dijalankan secara otomatis ketika tombol ditekan. Perhatikan contoh berikut :

- ```
1. tombol("Kembali", 100,80, 120, 40, menuUtama);
```

Ketika tombol "Kembali" ditekan, dengan kode di atas akan secara otomatis menjalankan fungsi menuUtama. Contoh penggunaan tombol dengan metode ini terdapat pada tutorial Bab 13.

Selanjutnya, untuk menambahkan interaktivitas pada aplikasi, pada baris 4 ditambahkan kode untuk mendeteksi tombol mouse yaitu :

- ```
4. canvas.onmouseup = mouseUp;
```

Seperti pada tutorial sebelumnya, kode ini membutuhkan fungsi mouseUp, yang akan dijalankan ketika tombol mouse di tekan dan dilepaskan (diklik). Selanjutnya pada fungsi tersebut, ditambahkan kode cekTombol untuk

mendeteksi adanya tombol yang diklik. Dalam contoh di atas, ketika salah satu tombol diklik, maka nilai dari variabel `teksLabel` akan diubah sesuai dengan label tombol yang ditekan.



Note : untuk menggunakan sebuah gambar *bitmap* sebagai tombol dapat digunakan fungsi sebagai berikut :

1. `tombolImg(label, x, y, l, t, data gambar);`

## B. Tutorial 4b - *Slider*

Elemen interaktif berikutnya dalam simulasi adalah *slider*. *Slider* digunakan untuk mengubah nilai suatu variabel secara dinamis dengan batasan minimal dan maksimal. Dalam beberapa contoh simulasi, *slider* digunakan untuk mengatur perubahan nilai variabel-variabel yang terlibat dalam suatu praktikum, sehingga pengguna dapat lebih mudah menyimpulkan hubungan dan akibat perubahan variabel yang ada terhadap luaran praktikum. Untuk membuat *slider* perhatikan contoh berikut :

### a. *File simulasi-4.js*

1. Lanjutkan mengedit *file simulasi-4.js*.
2. Perhatikan penambahan kode di bawah ini (ditandai dengan kotak), sehingga keseluruhan kode pada *simulasi-4.js* adalah sebagai berikut :

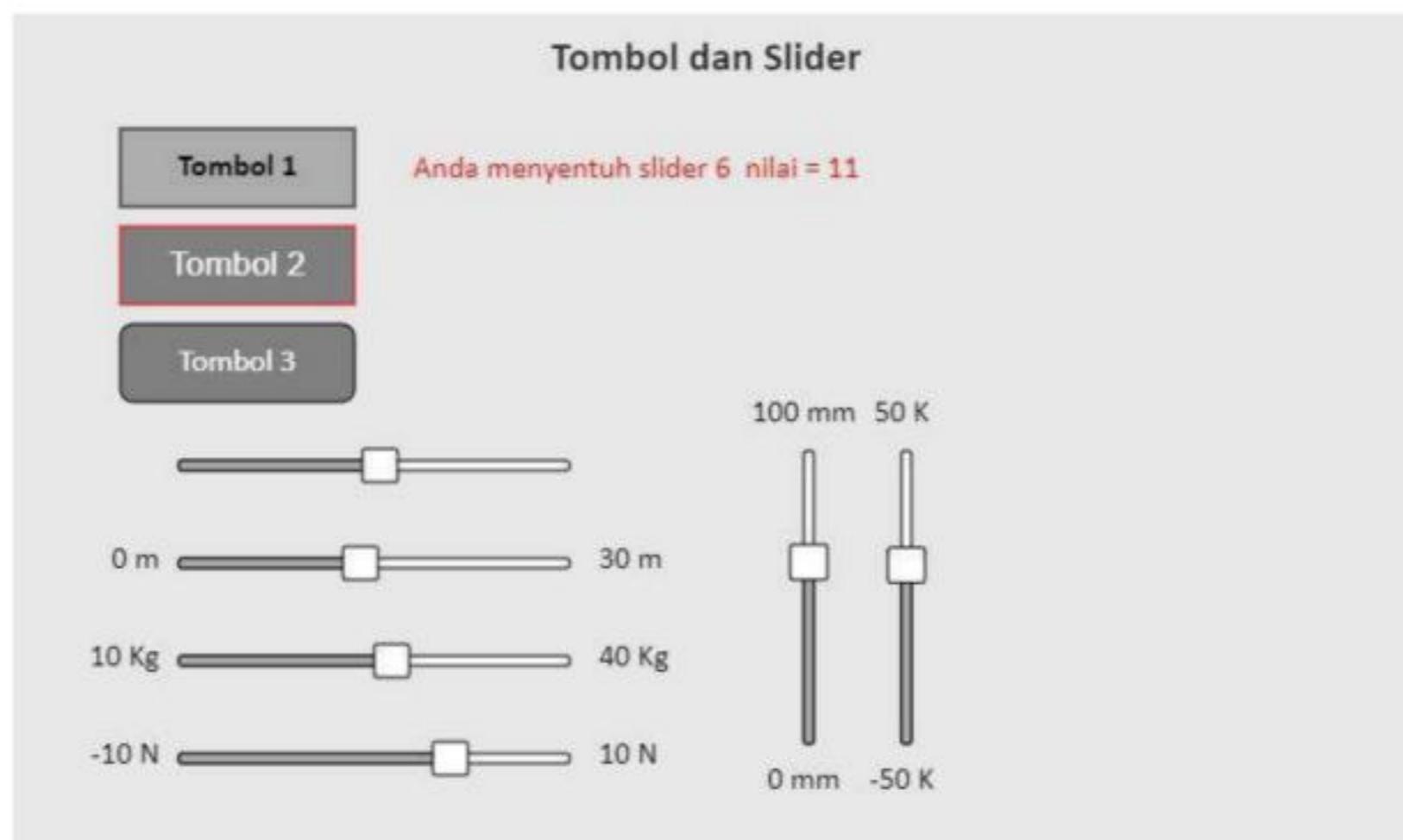
```
1. aturCanvas();
2. setJudul("Interaktivitas Dasar pada HTML Canvas");
3.
4. //listener untuk membaca event mouse
5. canvas.onmousedown = mouseDown;
6. canvas.onmouseup = mouseUp;
7.
8. var teksLabel = "Klik salah satu tombol";
9.
10. var slider1 = {tipe:"H", nama:"slider 1", x:100,y:250,
11. p:200, minS:0, maxS:100, vals:0, desimal:1, label:""}
12. var slider2 = {tipe:"H", nama:"slider 2", x:100,y:300,
13. p:200, minS:0, maxS:30, vals:10, desimal:1, label:"m"}
14. var slider3 = {tipe:"H", nama:"slider 3", x:100,y:350,
15. p:200, minS:10, maxS:40, vals:25, desimal:1,
16. label:"Kg"}
17. var slider4 = {tipe:"H", nama:"slider 4", x:100,y:400,
18. p:200, minS:-10, maxS:10, vals:0, desimal:1, label:"N"}
19. var slider5 = {tipe:"V", nama:"slider 5", x:450,y:225,
20. p:150, minS:0, maxS:100, vals:75, desimal:1,
21. label:"mm"}
22. var slider6 = {tipe:"V", nama:"slider 6", x:500,y:225,
23. p:150, minS:-50, maxS:50, vals:0, desimal:0, label:"K"}
```

```

17. function setSimulasi(){
18. hapusLayar("#e8e8e8");
19. teks("Tombol dan Slider", 0.5*(canvas.width), 50);
20. teks(teksLabel, 250, 105, "12pt Calibri", "red",
21. "left");
22. tombol("Tombol 1", 100,80, 120, 40);
23. tombol("Tombol 2", 100,130, 120, 40, "13pt Arial",
24. "white", "red", "gray");
25. tombol("Tombol 3", 100,180, 120, 40, "12pt Calibri",
26. "white", "black", "gray", "r");
27. //membuat slider
28. slider(slider1);
29. slider(slider2);
30. slider(slider3);
31. slider(slider4);
32. slider(slider5);
33. slider(slider6);
34. }
35. function mouseUp(event){
36. //prosedure mengecek tombol
37. var tombolAktif = cekTombol(event);
38. if (tombolAktif != ""){
39. teksLabel = "Anda menekan "+tombolAktif;
40. setSimulasi();
41. }
42. //menetralisir drag
43. canvas.onmousemove = null;
44. }
45. function mouseDown(event){
46. canvas.onmousemove = mouseDrag;
47. }
48.
49. function mouseDrag(event){
50. //prosedur mengecek slider
51. var sliderAktif = cekSlider(event);
52. if (sliderAktif != null){
53. teksLabel = "Anda menyentuh "+sliderAktif.nama+
54. nilai = "+sliderAktif.vals;
55. setSimulasi();
56. }
57.
58. setSimulasi();

```

3. Simpan *file* dan *refresh* internet *browser*, maka akan didapatkan hasil sebagai berikut :



Gambar 6.2 Hasil fungsi tombol dan *slider*

### b. Penjelasan Program

Fungsi *slider* akan menampilkan sebuah *slider* sesuai dengan data yang diinputkan. Pengaturan data dilakukan dengan mendeklarasikan variabel dengan parameter sebagai berikut :

|         |                                                                                                                                                                                                                                                                                                                                 |
|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Tipe    | Diisi dengan "H" untuk <i>slider</i> dengan arah horisontal<br>Atau "V" untuk <i>slider</i> dengan arah vertikal (tegak)                                                                                                                                                                                                        |
| nama    | Digunakan untuk mengidentifikasi nama dari <i>slider</i> .<br>Masing-masing <i>slider</i> harus memiliki nama yang unik agar tidak terjadi kesalahan pengecekan ketika fungsi <i>cekSlider</i> dijalankan                                                                                                                       |
| x       | Kordinat x dari awal <i>slider</i>                                                                                                                                                                                                                                                                                              |
| y       | Kordinat y dari awal <i>slider</i>                                                                                                                                                                                                                                                                                              |
| p       | Panjang <i>slider</i> dalam satuan pixel                                                                                                                                                                                                                                                                                        |
| minS    | Nilai minimal dari <i>slider</i>                                                                                                                                                                                                                                                                                                |
| maxS    | Nilai maksimal dari <i>slider</i>                                                                                                                                                                                                                                                                                               |
| vals    | Nilai dari <i>slider</i> . Dimana nilai <i>vals</i> secara otomatis tersimpan dan dapat dipanggil dengan kode <i>namaSlider.vals</i> sebagaimana pada baris 53.                                                                                                                                                                 |
| desimal | Nilai desimal dari <i>slider</i> . Dimana jika anda set 1, maka nilai <i>vals</i> memiliki detail 1 angka di belakang koma.                                                                                                                                                                                                     |
| label   | Merupakan label atau satuan yang akan ditampilkan di awal dan di akhir <i>slider</i> . Apabila anda tidak membutuhkan label dan angka minS dan maxS dapat anda ketikkan karakter kosong (""), atau jika anda menginginkan angka minS dan maxS muncul tanpa diikuti karakter lain, anda dapat menginputkan karakter spasi (" "). |

Selanjutnya untuk menampilkan *slider* ke canvas, digunakan fungsi *slider*, sebagaimana pada baris 26 sampai 31

```
26 slider(slider1);
```

*Slider* membutuhkan input *mouse* yang berbeda dengan tombol, dimana tombol hanya memerlukan klik *mouse*, sedangkan *slider* membutuhkan kondisi *mouse* ditekan dan digerakkan (*drag*), oleh karena itu pada baris 5 ditambahkan kode berikut :

```
5. canvas.onmousedown = mouseDown;
```

Untuk mendeteksi tombol mouse ditekan digunakan fungsi *mouseDown* (baris 45), yang di dalamnya mengaktifkan fungsi untuk mendeteksi gerakan *mouse*, yaitu fungsi *mouseDrag*. Pada fungsi *mouseDrag*, posisi *slider* akan di cek menggunakan kode *cekSlider* (baris 51), dan diupdate secara otomatis nilai *valS* nya berdasarkan posisi *slider*.

### C. Tutorial 5 – *Drag and Drop*

Elemen interaktif berikutnya yang sering digunakan dalam simulasi adalah kemampuan simulasi untuk *drag* objek (menggeser posisi objek dengan *mouse* atau *touch*). Dalam contoh ini akan dijelaskan proses menggerakkan gambar dengan *mouse* dalam format jangka sorong (*caliper*). Perhatikan langkah berikut :

#### a. *File HTML*

1. Seperti pada projek tutorial sebelumnya, buatlah sebuah *folder* khusus dengan nama **tutorial-5**. *Copy file tutorial-4.html* dan *paste* ke dalam *folder*.
2. *Rename* nama *file* menjadi **tutorial-5.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
12. <script src="simulasi-5.js"></script>
```

3. Simpan kembali *file* HTML.

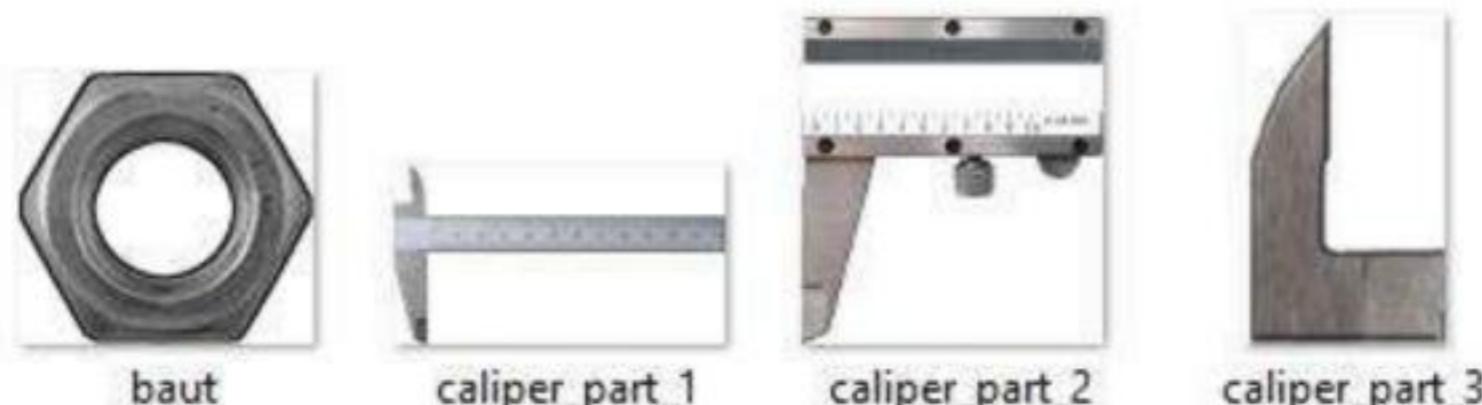
#### b. *File library dan Gambar*

1. *Copy paste* *file* **simulasiDasar.js** ke dalam *folder* **tutorial-5**.
2. Siapkan gambar bagian dari jangka sorong, anda dapat menggambar sendiri atau mengambil gambar dari *file* tutorial yang disertakan dalam buku.

Adapun *file* yang dimaksud bertipe PNG dengan *background* transparan, serta terdiri dari beberapa bagian yang terpisah (lihat gambar 6.3).

- Buatlah *folder images* di dalam *folder tutorial-5*, dan letakkan *file* gambar tersebut ke dalam *folder images*.

WANDAH > Book > physic > File Tutorial Buku > tutorial-5 > images



Gambar 6.3 *File* gambar bagian jangka sorong (*caliper*)

#### c. *File simulasi-5.js*

- Pada Notepad++, buatlah sebuah *file* baru

- Ketikan kode berikut :

```
1. aturCanvas();
2. setJudul("Interaktivitas Dasar pada HTML Canvas");
3.
4. //listener untuk membaca event mouse
5. canvas.onmousedown = mouseDown;
6. canvas.onmouseup = mouseUp;
7. hapusLayar("#e8e8e8");
8.
9. var xBase = 50;
10. var yBase = 100;
11. var xBaut = 600;
12. var yBaut = 500;
13. var xSnap = xBase+70; //70 (titik awal caliper)
14. var ySnap = yBase+275;
15. var isSnap = false;
16.
17. var fileGambar = {
18. caliper1: "images/caliper_part_1.png",
19. caliper2: "images/caliper_part_2.png",
20. caliper3: "images/caliper_part_3.png",
21. baut: "images/baut.png"
22. };
23.
24. var caliper = {nama:"caliper", x:xBase+70, y:yBase+80,
w:270, h:297, limit:"x"};
25. var baut = {nama:"baut", x:600, y:500, w:80, h:74,
limit:"xy"};
26. var xMove = 0;
```

```

27.
28. preload(fileGambar, setAwal);
29.
30. function setAwal(){
31. setDrag(caliper);
32. setDrag(baut);
33. setSimulasi();
34. }
35.
36. function setSimulasi(){
37. //hapus layar
38. hapusLayar();
39. //menampilkan teks
40. teks("Drag Objek", 0.5*(canvas.width), 50);
41.
42. //gambar part caliper
43. gambar(dataGambar.caliper3, xBase+5+xMove, yBase);
44. gambar(dataGambar.caliper1, xBase, yBase);
45. gambar(dataGambar.caliper2, xBase+70+xMove,
yBase+80);
46. gambar(dataGambar.baut, baut.x, baut.y);
47. }
48.
49. function mouseDown(event){
50. startDrag(mouseDrag);
51. }
52.
53. function mouseUp(event){
54. //menetralisir drag
55. canvas.onmousemove = null;
56. //menghitung posisi baut
57. if (jarak(baut.x, baut.y, xSnap, ySnap) < 50){
58. isSnap = true;
59. baut.x = xSnap;
60. baut.y = ySnap;
61. if (xMove < 80) {
62. xMove = 100;
63. caliper.x = xBase+170;
64. }
65. setSimulasi();
66. }else{
67. isSnap = false;
68. setSimulasi();
69. }
70. //mengembalikan posisi baut
71. if (!isSnap){
72. baut.x = xBaut;
73. baut.y = yBaut;
74. setSimulasi();
75. }
76. }

```

```

77.
78. function mouseDrag(event) {
79. //prosedur mengecek mode drag
80. var drag = cekDrag(event);
81. if (drag != null){
82. if (drag.nama == "caliper"){
83. xMove+=drag.dx;
84. if (isSnap){
85. if (xMove<baut.w) {
86. xMove = baut.w;
87. caliper.x = xBase+70+baut.w;
88. }
89. }else{
90. if (xMove < 0) {
91. xMove = 0;
92. caliper.x = xBase+70;
93. }
94. }
95. if (xMove > 350){
96. xMove = 350;
97. caliper.x = xBase+420;
98. }
99. }
100. setSimulasi();
101. }
102. }

```

3. Simpan file dan jalankan file **tutorial-5.html** melalui *internet browser*, maka akan didapatkan hasil sebagai berikut:



Gambar 6.4 Hasil tutorial *drag and drop*

#### d. Penjelasan Program

Pada tutorial di atas digunakan beberapa gambar untuk membentuk sebuah jangka sorong (*caliper*). Setelah gambar disiapkan dalam *folder images*, maka masing-masing gambar perlu diidentifikasi dengan cara memberikan nama pada variabel `siapkanGambar`. Sebagai contoh, gambar `baut.png` diberikan nama `baut`, dan untuk menggunakannya digunakan kode `dataGambar.baut` (baris 46).

Setelah gambar didefinisikan, maka perlu dipanggil melalui fungsi `preload`. Cara kerja *file* HTML dan JS adalah membuka *file* yang berasal dari *server* (dalam hal ini yang dimaksud dengan *server* adalah *file* lokal yang ada di *hard disk* kita). Proses membuka *file* tersebut dilakukan per *byte* data, sehingga membutuhkan waktu sepersekian detik. Apabila terdapat beberapa gambar yang akan digunakan, proses membuka *file* akan menjadi lebih lama. Sementara itu *file* kode yang berupa teks, biasanya memiliki ukuran yang lebih kecil jika dibandingkan *file* gambar. Dengan demikian kode dimungkinkan untuk terbaca lebih dahulu dan akan dijalankan oleh aplikasi, sementara *file* gambar belum terunduh secara sempurna dan belum teridentifikasi. Hal ini menyebabkan kesalahan program (*error*). Oleh karena itu, diperlukan sebuah fungsi untuk membuka *file* gambar secara utuh terlebih dahulu, dan aplikasi baru dijalankan setelah semua asset selesai diunduh. Di sinilah peran fungsi `preload`, file gambar akan diunduh seutuhnya dan akan menjalankan fungsi tertentu setelahnya. Adapun fungsi `preload` memiliki struktur sebagai berikut :

```
28. preload(file Gambar, fungsi setelah proses loading);
```

Pada baris 28, setelah seluruh *file* gambar terunduh, maka fungsi `setAwal` akan dijalankan.

Selanjutnya, untuk mengaktifkan fitur *drag*, diperlukan variabel yang memiliki parameter khusus. Pada tutorial di atas, terdapat 2 objek yang dapat di *drag* yaitu rahang sorong, dan baut, sehingga diperlukan 2 variabel pada baris 24 dan 25.

```
24. var caliper = {nama:"caliper", x:xBase+70, y:yBase+80,
w:270, h:297, limit:"x"};
```

Parameter yang diperlukan untuk membuat fitur *drag* adalah :

|       |                                                                                                                                                                                                                                  |
|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| nama  | Digunakan untuk mengidentifikasi nama dari objek <i>drag</i> . Masing-masing objek harus memiliki nama yang unik agar tidak terjadi kesalahan pengecekan ketika fungsi <code>cekDrag</code> dijalankan                           |
| x     | Kordinat x dari objek                                                                                                                                                                                                            |
| y     | Kordinat y dari objek                                                                                                                                                                                                            |
| w     | Lebar area objek <i>drag</i> dengan satuan <i>pixel</i>                                                                                                                                                                          |
| h     | Tinggi area objek <i>drag</i>                                                                                                                                                                                                    |
| limit | Batasan arah drag. Pada baris 24 misalnya, limit diatur dengan nilai "x", sehingga rahang sorong hanya dapat bergerak pada sumbu x.<br>Sedangkan pada baris 24, limit bernilai "xy" sehingga baut dapat bergerak ke segala arah. |

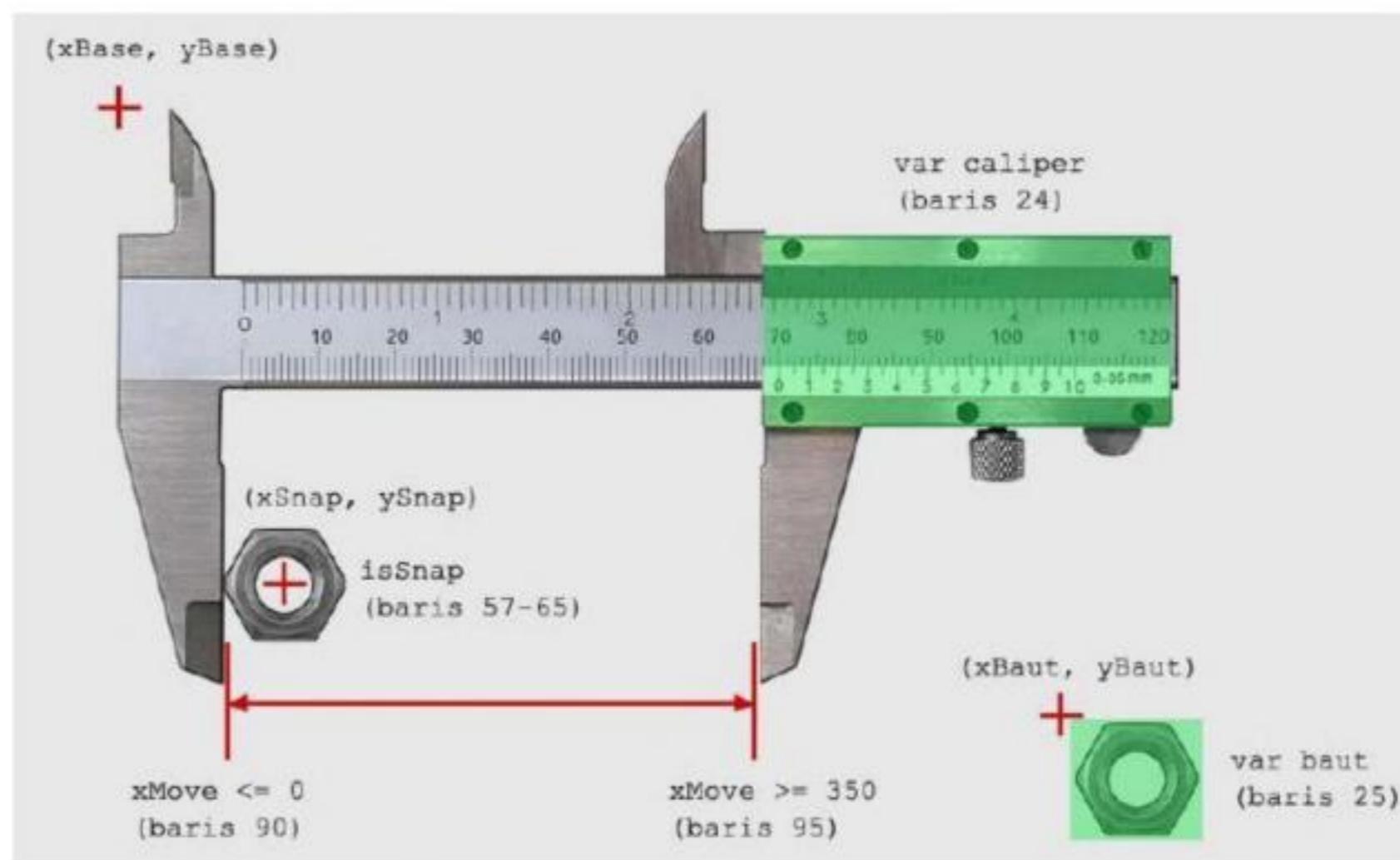
Pada baris 26 terdapat variabel `xMove`, yang digunakan untuk membatasi gerak rahang sorong (kode baris 83 – 96). Lihat penjelasan pada Gambar 6.5.

```
26. var xMove = 0;
```

Pada fungsi `setSimulasi`, untuk menampilkan gambar ke canvas digunakan fungsi `gambar`, dengan 3 parameter yaitu `dataGambar` yang akan ditampilkan, kordinat x dan kordinat y gambar.

```
43. gambar(dataGambar.caliper3, xBase+5+xMove, yBase);
```

Dengan menggunakan fungsi `mouseDrag`, posisi gambar `caliper2` (rahang sorong) dan baut `diupdate` sesuai dengan masukan dari pengguna, dan digambar ulang ke canvas dengan memanggil fungsi `setSimulasi` kembali (baris 100). Ketika pengguna melakukan *drag* objek, maka posisi baut dideteksi apakah mendekati posisi `snap` (`xSnap, ySnap`) menggunakan fungsi `jarak` (pada baris 57). Ketika *drag* dilepas dan baut dekat dengan posisi `snap`, maka baut akan "menempel" pada posisi yang ditentukan, dan nilai variabel `xMove` akan diatur ulang sehingga rahang sorong tidak dapat menutup melebihi lebar baut (baris 84 – 88).



Gambar 6.5 Penjelasan tutorial 5.

Dengan penambahan sedikit kode untuk menghitung lebar baut sampai 2 angka di belakang koma sebagaimana jangka sorong yang sebenarnya, dan variasi ukuran baut (objek lain), maka akan didapatkan sebuah simulasi pengukuran objek dengan jangka sorong virtual.

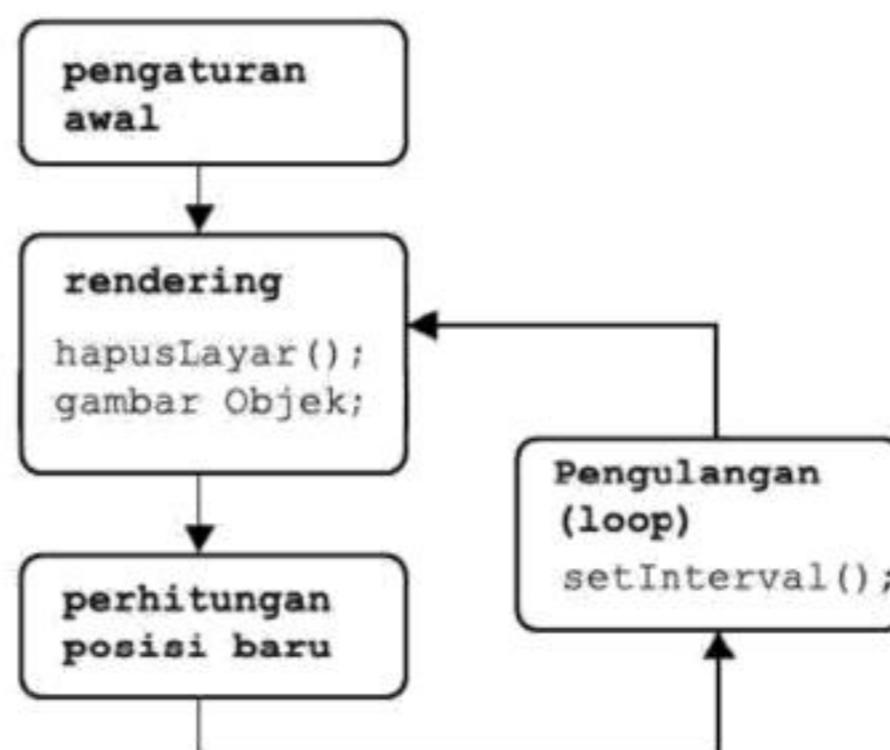


## BAB 7

### Animasi

#### 7.1 Konsep Dasar Animasi dengan Canvas

Salah satu upaya untuk menampilkan simulasi praktikum dalam laboratorium virtual adalah dengan cara menampilkan gerakan objek atau animasi. Animasi merupakan gambar bergerak yang berasal dari kumpulan berbagai objek yang disusun secara khusus, sehingga bergerak sesuai alur yang sudah ditentukan pada setiap hitungan waktu. Dalam HTML canvas, animasi berarti menggambar objek secara berulang ulang dengan durasi waktu tertentu. Adapun konsep dasar untuk menghasilkan sebuah animasi melalui HTML Canvas adalah sebagai berikut :



Gambar 7.1 Konsep animasi pada canvas

Animasi diawali dengan pengaturan awal, yang meliputi proses *preload* gambar, pengaturan variabel terkait dan pengaturan objek interaktif. Selanjutnya dilakukan proses *rendering*, yang diawali dengan menghapus canvas terlebih dahulu dan dikuti dengan menggambar seluruh objek. Berikutnya dilakukan perhitungan posisi baru dari objek yang dianimasikan, posisi disini bisa berarti perpindahan koordinat, perubahan sudut, perubahan gambar dan sebagainya. Agar gambar berubah sesuai dengan perhitungan, maka dilakukan pengulangan atau *loop* proses *rendering* menggunakan fungsi pengulangan seperti `setInterval` atau `setTimeout`.

## A. Tutorial 6 – Animasi roda gigi

### a. File HTML

1. Seperti pada projek tutorial sebelumnya, buatlah sebuah *folder* khusus dengan nama **tutorial-6**. *Copy file tutorial-5.html* dan *paste* ke dalam *folder*.
2. *Rename* nama *file* menjadi **tutorial-6.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
12. <script src="simulasi-6.js"></script>
```

3. Simpan kembali *file* HTML.

### b. File library

1. *Copy paste* *file* **simulasiDasar.js** ke dalam *folder* **tutorial-6**.

### c. File simulasi-6.js

1. Pada Notepad++, buatlah sebuah file baru
2. Ketikkan kode berikut :

```
1. aturCanvas();
2. setJudul("Animasi Roda Gigi");
3. hapusLayar("#e8e8e8");
4.
5. var sudut1 = 0;
6. var sudut2 = 17;
7. var sudut3 = 0;
8. var sudut4 = -12;
9.
10. var dataGear1 = {cx:150, cy:300, notches:10,
 radiusO:60, radiusI:50, radiusH:20, taperO:40,
 taperI:30}
11. var dataGear2 = {cx:322, cy:300, notches:20,
 radiusO:120, radiusI:110, radiusH:20, taperO:40,
 taperI:30}
12. var dataGear3 = {cx:600, cy:300, notches:30,
 radiusO:60, radiusI:50, radiusH:10, taperO:100,
 taperI:100, warnaIsi:"blue", warnaGaris:"black"}
13. var dataGear4 = {cx:684, cy:300, notches:15,
 radiusO:30, radiusI:20, radiusH:5, taperO:100,
 taperI:100, warnaIsi:"blue", warnaGaris:"black"}
14.
15. function setSimulasi(){
16. hapusLayar();
17. teks("Roda Gigi", 0.5*(canvas.width), 50);
18. //mengambar roda gigi
19. gear(dataGear1, sudut1);
20. gear(dataGear2, sudut2);
```

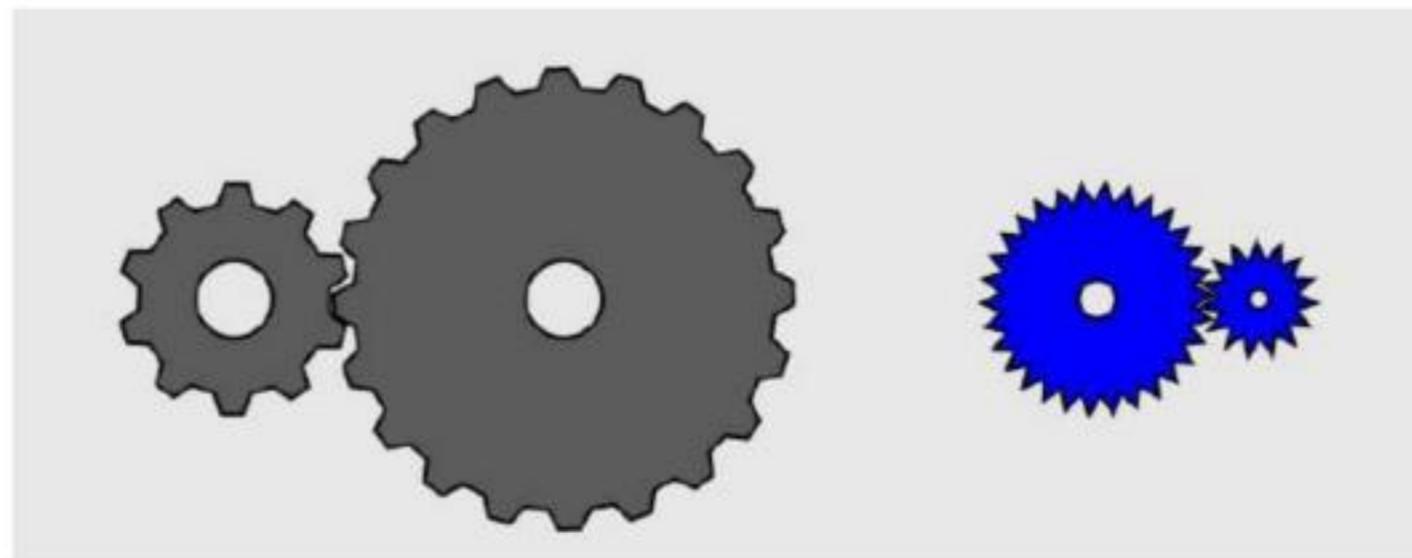
```

21. gear(dataGear3, sudut3);
22. gear(dataGear4, sudut4);
23. //perhitungan sudut
24. sudut1-=2;
25. sudut2+=1;
26. sudut3+=1;
27. sudut4-=2;
28. }
29.
30. function jalankanSimulasi() {
31. setSimulasi();
32. window.setTimeout(jalankanSimulasi, 30);
33. }
34.
35. jalankanSimulasi();

```

3. Simpan dengan nama **simulasi-6.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-6.

Jalankan *file* **tutorial-6.html**, dengan cara membukanya di *internet browser*. Maka kode di atas akan menampilkan animasi roda gigi berputar sebagai berikut :



Gambar 7.2 Animasi roda gigi

#### d. Penjelasan Program

Sesuai dengan diagram pada Gambar 7.1, proses animasi diawali dengan mengatur variabel-variabel yang akan digunakan untuk menggambar roda gigi dan mengatur sudut putar dari roda gigi.

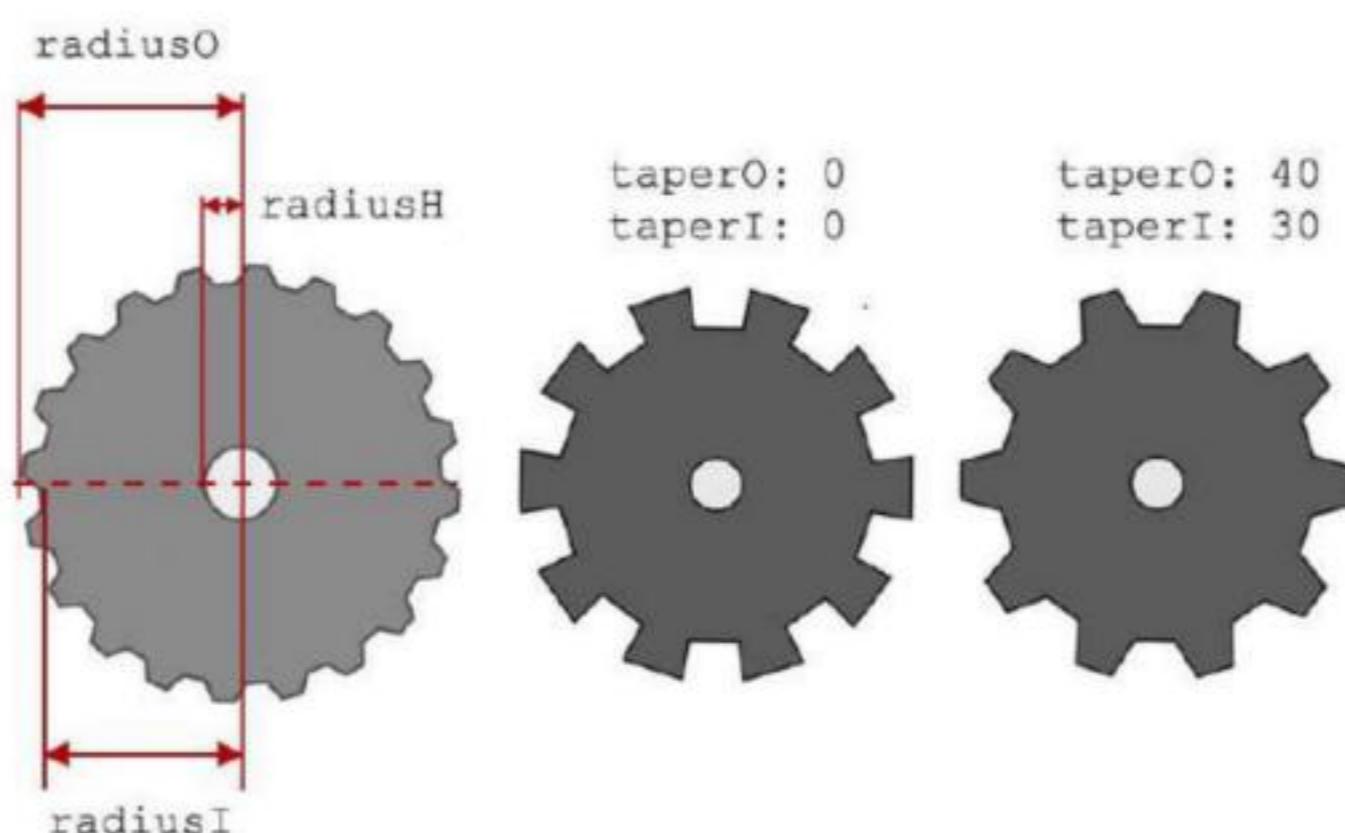
Dalam *library* **simulasiDasar.js**, menggambar roda gigi memerlukan pengaturan objek dengan beberapa parameter seperti pada variabel **dataGear1**.

---

|                |                                 |
|----------------|---------------------------------|
| <b>cx</b>      | Kordinat x dari pusat roda gigi |
| <b>cy</b>      | Kordinat y dari pusat roda gigi |
| <b>notches</b> | Jumlah gigi                     |
| <b>radiusO</b> | Jari-jari luar roda gigi        |

---

|            |                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| radiusI    | Jari-jari dalam roda gigi                                                                                                                                                                                                                                                                                                                                                                                                     |
| radiusH    | Jari-jari lubang tengah roda gigi                                                                                                                                                                                                                                                                                                                                                                                             |
| taperO     | Prosentase ukuran gigi luar.<br>Pada variabel <code>dataGear1</code> dan <code>dataGear2</code> , <code>taperO</code> bernilai 40, hal ini berarti lebar gigi luar mengerucut sebesar 40%. Sedangkan pada <code>dataGear3</code> dan <code>dataGear4</code> , <code>taperO</code> bernilai 100 sehingga mengerucut sempurna dan membentuk gigi lancip.                                                                        |
| taperI     | Prosentase ukuran gigi dalam.<br>Pada variabel <code>dataGear1</code> dan <code>dataGear2</code> , <code>taperI</code> bernilai 30, hal ini berarti lebar gigi dalam melebar sebesar 30%, memberikan jarak 30% antara gigi satu dengan gigi lainnya. Sedangkan pada <code>dataGear3</code> dan <code>dataGear4</code> , <code>taperI</code> bernilai 100 sehingga melebar sempurna dan saling menyambung dengan gigi lainnya. |
| warnalsi   | Warna dari roda gigi. Secara <i>default</i> jika tidak didefinisikan akan menghasilkan warna abu-abu gelap.                                                                                                                                                                                                                                                                                                                   |
| warnaGaris | Warna garis tepi dari roda gigi. Secara <i>default</i> jika tidak didefinisikan akan menghasilkan warna hitam.                                                                                                                                                                                                                                                                                                                |



Gambar 7.3 Parameter data untuk fungsi `gear`.

Setelah 4 buah variabel data untuk roda gigi didefinisikan, tahap berikutnya adalah menggambar semua objek di layar. Sesuai dengan Gambar 7.1, proses tersebut diawali dengan fungsi `hapusLayar`, agar gambar tidak saling menumpuk. Selanjutnya gambar roda gigi digambar dengan fungsi `gear`.

```
19. gear(dataGear1, sudut1);
```

Selanjutnya variabel `sudut1`, diubah dengan cara dikurangi 2, setiap langkah. Dengan demikian, pada penggambaran (*rendering*) berikutnya, roda gigi 1 akan berputar 2 derajat berlawanan arah jarum jam. Demikian halnya dengan roda

gigi 2-4, dilakukan perubahan sudut dengan cara menambah atau mengurangi sudutnya.

Untuk menganimasikan roda gigi, fungsi `setSimulasi` perlu dijalankan secara berulang-ulang (*loop*). Maka, digunakan fungsi `jalankanSimulasi`, yang di dalamnya terdapat perintah `setTimeout`.

```
32. window.setTimeout(jalankanSimulasi, 30);
```

Fungsi `setTimeout`, digunakan untuk mengatur eksekusi sebuah fungsi pada waktu tertentu (dalam satuan milidetik) sebanyak 1 kali. Pada contoh di atas, kode akan menunggu sampai dengan 30 milidetik sebelum menjalankan fungsi `jalankanSimulasi`. Setelah 30 milidetik, fungsi `jalankanSimulasi` dijalankan maka fungsi-fungsi di dalamnya akan dieksekusi, dimulai dengan fungsi `setSimulasi`, yang berarti gambar di canvas akan dibuat ulang dengan nilai variabel `sudut1-4` yang sudah berubah. Setelah itu pada baris berikutnya (baris 32), terdapat fungsi `setTimeout`, sehingga menunggu lagi selama 30 milidetik. Proses ini berulang-ulang sehingga menghasilkan efek roda gigi yang bergerak (animasi).

## 7.2 Animasi *Sprite*

Dalam beberapa kasus simulasi, diperlukan sebuah animasi yang detail dan kompleks. Sebagai contoh dalam simulasi biologi sistem peredaran darah, maka diperlukan gambar dinamis dengan gerakan yang sangat kompleks, atau dalam simulasi mesin dibutuhkan gerakan kompleks mulai dari gerak piston, katup, representasi bahan bakar dan gas buang dan seterusnya. Animasi kompleks tersebut memungkinkan dibuat dengan kode, namun kurang efisien. Para pengembang media pembelajaran atau simulasi, lebih memilih menampilkan animasi kompleks dalam bentuk video, file GIF/PNG *animation* atau animasi *sprite*.

Animasi *sprite*, merupakan animasi yang disusun oleh *sprite* yaitu beberapa gambar objek animasi yang disatukan dalam gambar berukuran lebih besar, dan dapat ditampilkan per bagian sesuai dengan kebutuhan. Penggunaan *sprite* secara umum dikenal dalam pengembangan permainan digital (*game*). Penggunaan *sprite* memiliki keuntungan yaitu ukuran *file* yang lebih kecil, serta pengaturan yang relatif lebih mudah. Untuk memahami konsep animasi *sprite*, perhatikan contoh berikut :



Gambar 7.4 *Sprite* mesin

(sumber : [https://en.wikipedia.org/wiki/File:4StrokeEngine\\_Ortho\\_3D.gif](https://en.wikipedia.org/wiki/File:4StrokeEngine_Ortho_3D.gif))

Pada contoh di atas, animasi sebuah mesin bakar dibuat secara *frame by frame* yang terdiri atas 20 gambar. Pada tiap-tiap gambar terdapat gerakan mesin yang bertahap dan berulang. 20 Gambar tersebut disatukan menjadi satu kesatuan gambar dengan nama animasi\_mesin.png, yang mana dengan fungsi *sprite* pada simulasiDasar.js dapat ditampilkan satu bagian yang dibutuhkan saja.

#### A. Tutorial 7 – Animasi *sprite*

##### a. *File HTML*

1. Seperti pada projek tutorial sebelumnya, buatlah sebuah *folder* khusus dengan nama **tutorial-7**. *Copy* file **tutorial-6.html** dan *paste* ke dalam *folder*.
2. *Rename* nama *file* menjadi **tutorial-7.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
12. <script src="simulasi-7.js"></script>
```

3. Simpan kembali file HTML.

##### b. *File library dan file gambar*

1. *Copy paste* file **simulasiDasar.js** ke dalam *folder* **tutorial-7**.

2. Tambahkan *file* gambar animasi mesin yang telah disusun dalam bentuk sprite, yaitu **animasi\_mesin.png** yang terdapat pada *file* tutorial penyerta buku. *Copy paste* ke *folder* tutorial-7.



Gambar 7.5 File animasi\_mesin.png

(referensi : [https://en.wikipedia.org/wiki/File:4StrokeEngine\\_Ortho\\_3D.gif](https://en.wikipedia.org/wiki/File:4StrokeEngine_Ortho_3D.gif))

*File* di atas memiliki ukuran 900x960 pixel, dengan masing-masing *sprite* berukuran 180x240 pixel.

### c. *File* simulasi-7.js

1. Pada Notepad++, buatlah sebuah *file* baru
2. Ketikkan kode berikut :

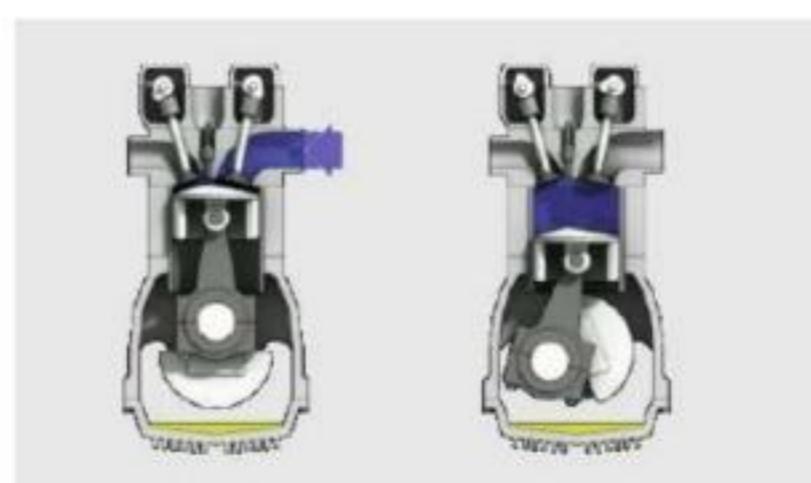
```
1. aturCanvas();
2. setJudul("Animasi Mesin dengan Teknik Sprite");
3. hapusLayar("#e8e8e8");
4.
5. var fileGambar = {
6. animasi: "images/animasi_mesin.png"
7. };
8.
9. var mesin = {imgW:180, imgH:240}
10.
11. preload(fileGambar, setAwal);
12.
13. function setAwal(){
14. mesin.img = dataGambar.animasi;
15. jalankanSimulasi();
16. }
17.
18. function setSimulasi(){
19. hapusLayar();
20. teks("Animasi Mesin", 0.5*(canvas.width), 50);
21. //menggambar sprite mesin
22. sprite(mesin, 100, 100, 1);
23. loopSprite(mesin, 300, 100);
```

```
24. }
25.
26. function jalankanSimulasi() {
27. setSimulasi();
28. window.setTimeout(jalankanSimulasi, 90);
29. }
```

3. Simpan dengan nama **simulasi-7.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-7.

Jalankan *file* **tutorial-7.html**, dengan cara membukanya di *internet browser*.

Maka kode di atas akan menampilkan animasi mesin sebagai berikut :



Gambar 7.6 Hasil tutorial animasi *sprite*

#### d. Penjelasan Program

Untuk menggunakan teknik *sprite*, perlu didefinisikan terlebih dahulu ukuran *sprite*, yang dalam hal ini ukuran gambar setiap mesin adalah 180 x 240 *pixel*.

```
9. var mesin = {imgW:180, imgH:240}
```

Selanjutnya, gambar yang telah dibuka dengan fungsi *preload*, perlu didefinisikan pula ke dalam variabel *mesin*.

```
14. mesin.img = dataGambar.animasi;
```

Setelah itu terdapat dua cara untuk menampilkan *sprite*, yaitu menampilkan sebuah *frame* (gambar tunggal) dan menampilkan animasi *loop*. Untuk menampilkan *frame* tunggal, digunakan struktur sebagai berikut :

```
22. sprite(data sprite, x, y, frame);
```

Sedangkan untuk menampilkan animasi *sprite* secara *loop*, digunakan fungsi sebagai berikut :

```
23. loopSprite(data sprite, x, y);
```

## BAB 8

### Penjumlahan Vektor

#### 8.1 Menentukan Penjumlahan Vektor

Setelah memahami teknik dasar dalam pembuatan simulasi, maka pada bab ini secara khusus membahas penerapannya dalam bentuk simulasi. Tema pertama yang diambil adalah simulasi penjumlahan vektor dengan teknik grafik/analitik (menggunakan koordinat kartesius).

Besaran vektor merupakan besaran yang mempunyai nilai (besar) dan arah. Contoh besaran vektor, antara lain, perpindahan, kecepatan, percepatan, momentum, dan gaya. Untuk menyatakan besaran vektor, harus menggunakan nilai (angka) dan arah tertentu. Sebuah vektor digambarkan oleh sebuah anak panah. Panjang anak panah mewakili besar atau nilai vektor, sedangkan arah anak panah mewakili arah vektor. Notasi atau simbol sebuah vektor dapat menggunakan satu atau dua huruf dengan tanda panah di atasnya, misalnya  $\vec{A}$  atau  $\overrightarrow{AB}$ . Dalam contoh simulasi yang akan kita buat, vektor tersebut digambarkan dalam bentuk panah berwarna, untuk membedakan antara vektor  $\vec{A}$ , vektor  $\vec{B}$  dan vektor  $\overrightarrow{AB}$ .

Untuk menghitung penjumlahan antara 2 vektor, kita dapat melakukannya dengan beberapa metode. Salah satu metode dapat digunakan adalah metode analitik, yaitu menentukan besar dan arah vektor melalui rumus dan sketsa. Metode ini dilakukan menggunakan acuan berupa sistem koordinat kartesius dengan titik pangkal di koordinat  $(0, 0)$ . Sebuah vektor  $\vec{A}$  yang diuraikan menjadi dua buah vektor komponen, masing-masing berada pada sumbu-x dan sumbu-y.  $A_x$  adalah komponen vektor  $\vec{A}$  pada sumbu-x dan  $A_y$  adalah komponen vektor  $\vec{A}$  pada sumbu-y, sehingga besar setiap komponennya adalah :

$$A_x = A \sin \theta \text{ dan } A_y = A \cos \theta$$

Adapun langkah yang perlu dilakukan untuk menghitung jumlah vektor  $\vec{A}$  dan vektor  $\vec{B}$  adalah sebagai berikut :

1. Gambar koordinat kartesius x-y.
2. Letakkan titik tangkap semua vektor pada titik asal  $(0,0)$ .

3. Tentukanlah resultan vektor-vektor komponen pada setiap sumbu, misalnya:

$\Sigma R_x$  = vektor-vektor komponen pada sumbu-x.

$\Sigma R_y$  = vektor-vektor komponen pada sumbu-y.

4. Besar vektor resultannya dapat dihitung menggunakan rumus Pitagoras :

$$R = \sqrt{(\Sigma R_x)^2 + (\Sigma R_y)^2}$$

Dengan langkah tersebut, selanjutnya kita dapat menjabarkannya dalam bentuk kode Javascript. Perhatikan tutorial-8 berikut

#### A. Tutorial 8 – Penjumlahan Vektor

##### a. File HTML

1. Seperti pada projek tutorial sebelumnya, buatlah sebuah *folder* khusus dengan nama **tutorial-8**. *Copy file tutorial-7.html* dan *paste* ke dalam *folder*.
2. *Rename* nama *file* menjadi **tutorial-8.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
12. <script src="simulasi-8.js"></script>
```

3. Simpan kembali *file* HTML.

##### b. File library

1. *Copy paste* *file* **simulasiDasar.js** ke dalam *folder* **tutorial-8**.

##### c. File simulasi-8.js

1. Pada Notepad++, buatlah sebuah *file* baru

2. Ketikan kode berikut :

```
1. aturCanvas();
2. setJudul("Resultan Vektor");
3. hapusLayar("#b3cfe5");
4.
5. //listener untuk membaca event mouse
6. canvas.onmousedown = mouseDown;
7. canvas.onmouseup = mouseUp;
8.
9. //membuat grafik kordinat
10. var graf = {startX:80, startY:150, dataW:10, dataH:10,
tileW:35, skalaX:4, skalaY:4, desimalX:0, desimalY:0,
offsetX:5, offsetY:5, xLabel:'x (cm)', yLabel:'y (cm)',
fontLabel:'12pt Calibri', warnaBG:'#daf6fb',
warnaGaris:'#000', warnaLabel:'#000'}
```

```

11. var slider1 = {tipe:"H", nama:"slider 1", x:550,y:400,
12. p:150, minS:0, maxS:10, valS:8, desimal:1, label:" "}
13. var slider2 = {tipe:"H", nama:"slider 2", x:550,y:450,
14. p:150, minS:0, maxS:10, valS:6, desimal:1, label:" "}
15. var slider3 = {tipe:"H", nama:"slider 3", x:550,y:500,
16. p:150, minS:0, maxS:360, valS:0, desimal:0, label:"0"}
17. var slider4 = {tipe:"H", nama:"slider 4", x:550,y:550,
18. p:150, minS:0, maxS:360, valS:90, desimal:0, label:"0"}
19. var vApanjang = 8.0;
20. var vBpanjang = 6.0;
21. var vRpanjang = 0.0;
22. var vAsudut = 0.0;
23. var vBsudut = 90.0;
24. var vRsudut = 0.0;
25. var vlx, vly, v2x, v2y, vrax, vry;
26. function setSimulasi(){
27. //hapus layar
28. hapusLayar();
29. //menampilkan teks
30. teks("Resultan Vektor", 0.5*(canvas.width), 40, 'bold
31. 18pt Calibri', 'blue', 'center');
32. teks("Simulasi menunjukkan dua vektor dan penjumlahannya,
33. atau disebut sebagai resultan vektor",0.5*(canvas.width),
34. 75, "12pt Calibri", "#000", "center");
35. grafik(graf);
36. teks("Vektor A = "+vApanjang, 550, 380, "bold 14pt
37. Calibri", "#000", "left");
38. slider(slider1);
39. teks("Vektor B = "+vBpanjang, 550, 430, "bold 14pt
40. Calibri", "#000", "left");
41. slider(slider2);
42. teks("Sudut Vektor A = "+vAsudut+"0", 550, 480, "bold 14pt
43. Calibri", "#000", "left");
44. slider(slider3);
45. teks("Sudut Vektor B = "+vBsudut+"0", 550, 530, "bold 14pt
46. Calibri", "#000", "left");
47. slider(slider4);
48. //gambar vektor
49. var xPusat = graf.startX+graf.offsetX*graf.tileW;
50. var yPusat = graf.startY+graf.offsetY*graf.tileW;
51. vlx = vApanjang*Math.cos(vAsudut*Math.PI/180);
52. vly = vApanjang*Math.sin(vAsudut*Math.PI/180);

```

```

53. vRpanjang=Math.sqrt(vrx*vrx+vry*vry);
54. //gambar komponen penyusun vektor
55. panah(xPusat,yPusat,2*v1x,0,5,4,"#ffd0d0");
56. panah(xPusat+10*v1x,yPusat,0,2*v1y,5,4,"#ffd0d0");
57.
58. panah(xPusat+10*v1x,yPusat-10*v1y,2*v2x,0,5,4,"#8adaff");
59. panah(xPusat+10*v1x,yPusat-10*v1y,0,2*v2y, 5,4,
60. "#8adaff");
61. panah(xPusat,yPusat,2*vrx,0,5,4,"#acf1a3");
62. panah(xPusat+10*vrx,yPusat,0,2*vry,5,4,"#acf1a3");
63.
64. panah(xPusat,yPusat,2*v1x,2*v1y,5,4,"red");
65. panah(xPusat+10*v1x,yPusat-10*v1y,2*v2x,2*v2y,5,4,
66. "blue");
67. panah(xPusat,yPusat,2*vrx,2*vry,5,4,"#28db10");
68. //tampilkan data angka
69. teks("Panjang", 580, 130, 'Bold 15pt Calibri', '#313131',
70. 'left');
71. teks("X", 700, 130, 'Bold 15pt Calibri', '#313131');
72. teks("Y", 750, 130, 'Bold 15pt Calibri', '#313131');
73. //vektor A
74. panah(530, 160, 40, 0, 1, 2, 'red');
75. teks("V A = "+vApanjang.toFixed(2), 580, 160, 'Bold 13pt
76. Calibri', '#990020', 'left');
77. teks(v1x.toFixed(2), 700, 160, 'Bold 13pt Calibri',
78. '#990020');
79. teks(v1y.toFixed(2), 750, 160, 'Bold 13pt Calibri',
80. '#990020');
81. teks(v2x.toFixed(2), 700, 190, 'Bold 13pt Calibri',
82. '#004fb0');
83. teks(v2y.toFixed(2), 750, 190, 'Bold 13pt Calibri',
84. '#004fb0');
85. teks(vrz.toFixed(2), 700, 220, 'Bold 13pt Calibri',
86. '#007200');
87. }
88.
89. function mouseDown(event){
90. canvas.onmousemove = mouseDrag;
91. }
92.
93. function mouseUp(event){
94. canvas.onmousemove = null;

```

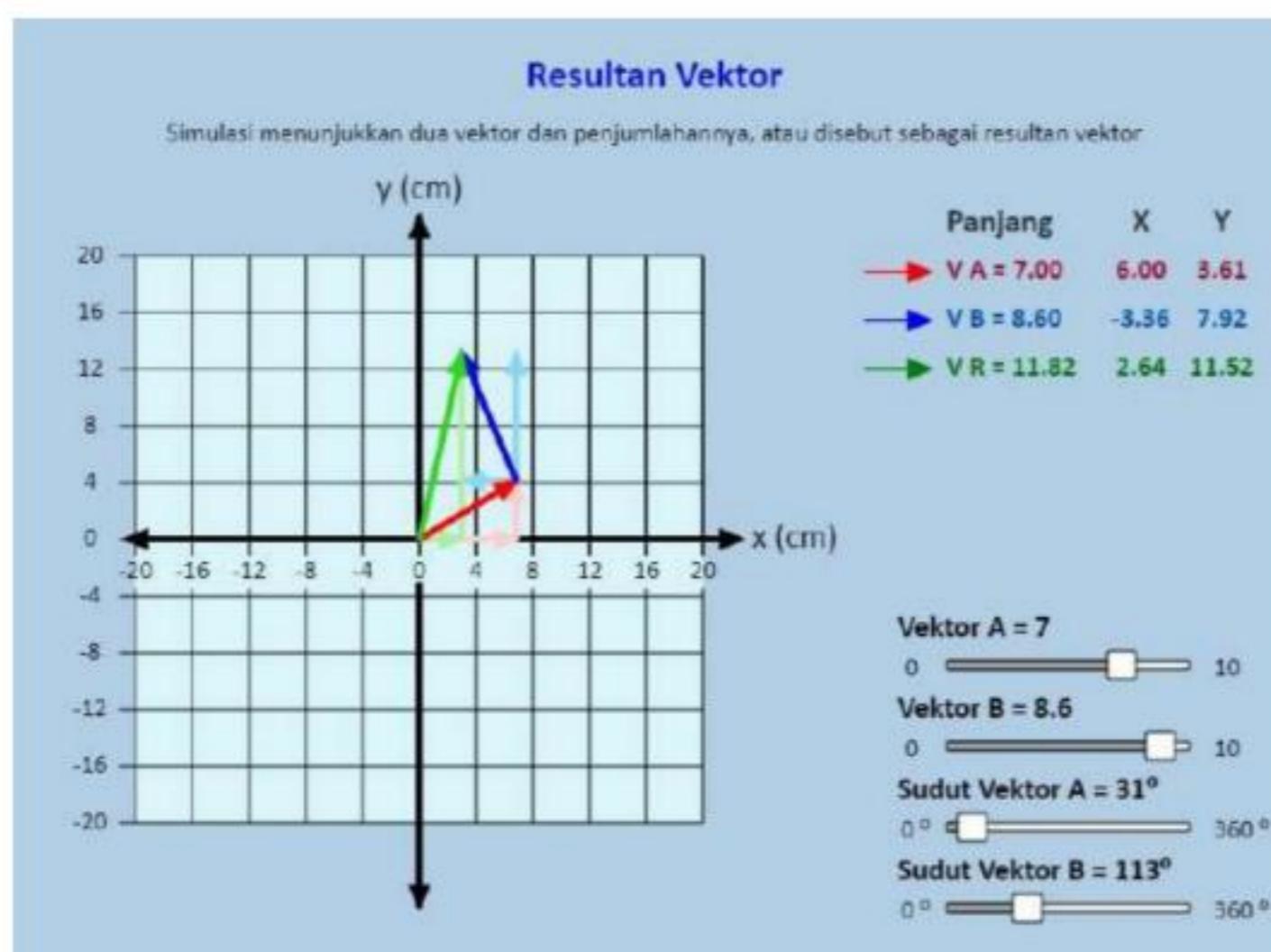
```

95. }
96. function mouseDrag(event) {
97. //prosedur mengecek slider
98. var sliderAktif = cekSlider(event);
99. if (sliderAktif != null){
100. if (sliderAktif.nama == "slider 1") {
101. vApanjang = Number(sliderAktif.vals);
102. }
103. if (sliderAktif.nama == "slider 2") {
104. vBpanjang = Number(sliderAktif.vals);
105. }
106. if (sliderAktif.nama == "slider 3") {
107. vAsudut = Number(sliderAktif.vals);
108. }
109. if (sliderAktif.nama == "slider 4") {
110. vBsudut = Number(sliderAktif.vals);
111. }
112. setSimulasi();
113. }
114. }
115.
116. setSimulasi();

```

3. Simpan dengan nama **simulasi-8.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-8.

Jalankan *file* **tutorial-8.html**, dengan cara membukanya di *internet browser*. Maka kode di atas akan menghasilkan simulasi penjumlahan vektor sebagai berikut :



Gambar 8.1 Hasil tutorial penjumlahan vektor

#### d. Penjelasan Program

Sesuai dengan langkah menghitung resultan 2 vektor yang telah dijelaskan di atas, pertama-tama dibutuhkan sebuah kordinat kartesius yang didefinisikan pada variabel *graf* (baris 10). Selanjutnya vektor  $\vec{A}$  dan vektor  $\vec{B}$  didefinisikan besaran dan arah nya melalui variabel *vApanjang* dan *vAsudut* (baris 16-22).

Untuk perhitungan vektor, dilakukan dengan trigonometri sederhana dimana sudut yang dibentuk perlu dikalikan dengan  $\pi \times 180$ . Sedangkan penjumlahan vektor dilakukan dengan rumus pitagoras (baris 53).

```
47. v1x = vApanjang*Math.cos(vAsudut*Math.PI/180);
48. v1y = vApanjang*Math.sin(vAsudut*Math.PI/180);
49. v2x = vBpanjang*Math.cos(vBsudut*Math.PI/180);
50. v2y = vBpanjang*Math.sin(vBsudut*Math.PI/180);
51. vrX = v1x + v2x;
52. vrY = v1y + v2y;
53. vRpanjang=Math.sqrt(vrX*vrX+vrY*vrY);
```

Pada akhirnya untuk menggambar vektor digunakan fungsi panah:

```
panah(xAwal, yyAwal, panjang sumbu x, panjang sumbu y,
satuan skala, ketebalan garis, warna garis);
```

Untuk mengubah nilai variabel vektor yang ada, digunakan fitur slider dan dilakukan proses *update* setiap kali terjadi perubahan pada nilai *vals* pada slider dengan memanggil kembali fungsi *setSimulasi()* (baris 112).

## BAB 9

### Simulasi Gerak

#### 9.1 Gerak Lurus

Gerak adalah perubahan posisi suatu objek yang diamati dari suatu titik acuan. Titik acuan yang dimaksud didefinisikan sebagai titik awal objek tersebut ataupun titik tempat pengamat berada. Benda yang bergerak memiliki kecepatan yang merupakan jarak perpindahan dibagi waktu tempuhnya. Benda bergerak juga memiliki percepatan yang didefinisikan sebagai perubahan kecepatan tiap satuan waktu.

Hubungan antara gerak, waktu, kecepatan dan percepatan dapat disimulasikan dalam beberapa bentuk. Dalam laboratorium fisika simulasi pada umumnya membutuhkan lintasan, beban dan mesin waktu ketik (*time ticker*) yang pada akhirnya dapat disimpulkan apakah gerak benda merupakan gerak lurus beraturan atau gerak lurus berubah beraturan. Dalam membangun simulasi mengenai gerak lurus, maka diperlukan beberapa persamaan terkait dengan variabel-variabel yang terlibat sebagai berikut :

$$v_t = v_0 + a \cdot t$$
$$s = v_0 \cdot t + (a \cdot t \cdot t) / 2$$

Dimana  $V_t$  = Kecepatan akhir,  $V_0$  = Kecepatan awal,  $a$  = percepatan,  $s$  = jarak dan  $t$  = waktu. Selanjutnya variabel-variabel yang terlibat akan diatur dan disesuaikan ke dalam bentuk kode Javascript. Pada tutorial berikut, akan disimulasikan gerak antara 2 benda (mobil) sebagai pengganti troli yang secara umum digunakan dalam praktik laboratorium. Tujuan dari simulasi ini adalah untuk mengetahui hubungan antara variabel kecepatan, percepatan dan jarak.

#### A. Tutorial 9 – Kecepatan antara 2 Objek

##### a. File HTML

1. Seperti pada projek tutorial sebelumnya, buatlah sebuah *folder* khusus dengan nama **tutorial-9**. *Copy file tutorial-8.html* dan *paste* ke dalam *folder*.

2. *Rename* nama *file* menjadi **tutorial-9.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut:

12.     `<script src="simulasi-9.js"></script>`

3. Simpan kembali *file* HTML.

**b. File library dan file gambar**

1. *Copy paste file* **simulasiDasar.js** ke dalam *folder* **tutorial-9**.
2. Buatlah sebuah *folder* **images** kemudian salin *file* gambar mobil merah dan mobil biru yang terdapat pada *folder* tutorial.

**c. File simulasi-9.js**

1. Pada Notepad++, buatlah sebuah *file* baru

2. Ketikan kode berikut :

```
1. aturCanvas();
2. setJudul("Gerak dan Kecepatan");
3. hapusLayar("#b3cfe5");
4.
5. //listener untuk membaca event mouse
6. canvas.onmousedown = mouseDown;
7. canvas.onmouseup = mouseUp;
8.
9. var graf = {startX:50, startY:140, dataW:16, dataH:3,
 tileW:40, skalaX:10, skalaY:10, desimalX:0, desimalY:0,
 offsetX:0, offsetY:3, xLabel:'x (m)', yLabel:'noaxis',
 fontLabel:'11pt Calibri', warnaBG:'#daf6fb',
 warnaGaris:'#000', warnaLabel:'#000'}
10.
11. var slider1 = {tipe:"H", nama:"posisi1", x:75,y:420, p:200,
 minS:0, maxS:50, vals:0, desimal:0, label:"m"}
12. var slider2 = {tipe:"H", nama:"kecepatan1", x:75,y:480,
 p:200, minS:0, maxS:10, vals:5, desimal:1, label:"m/s"}
13. var slider3 = {tipe:"H", nama:"acc1", x:75,y:540, p:200,
 minS:-2, maxS:2, vals:0, desimal:1, label:"m/s2"}
14. var slider4 = {tipe:"H", nama:"posisi2", x:475,y:420,
 p:200, minS:0, maxS:50, vals:0, desimal:0, label:"m"}
15. var slider5 = {tipe:"H", nama:"kecepatan2", x:475,y:480,
 p:200, minS:0, maxS:10, vals:5, desimal:1, label:"m/s"}
16. var slider6 = {tipe:"H", nama:"acc2", x:475,y:540, p:200,
 minS:-2, maxS:2, vals:0, desimal:1, label:"m/s2"}
17.
18. var index = 0;
19. var xBase = graf.startX;
20. var yBase = graf.startY;
21. var radius = 6;
22. var x1Init = 0;
23. var x1 = xBase;
```

```

24. var v1 = 5.0;
25. var a1 = 0.0
26. var y1 = yBase + graf.tileW;
27. var x2Init = 0.0;
28. var x2 = xBase;
29. var v2 = 5.0;
30. var a2 = 0.0
31. var y2 = yBase + graf.tileW*2;
32. var time = 0.0;
33. var timer;
34. var simAktif = 0;
35.
36. var fileGambar = {
37. mobil_biru: "images/mobil_biru.png",
38. mobil_merah: "images/mobil_merah.png",
39. }
40.
41. preload(fileGambar, setSimulasi);
42.
43. function setSimulasi(){
44. //hentikan aplikasi jika mobil mencapai ujung lintasan
45. if ((x1 >= (xBase+graf.tileW*graf.dataW)) || (x2 >=
(xBase+graf.tileW*graf.dataW)) || (x1 < xBase) || (x2 < xBase) || (time >= 50)) simAktif = 0;
46. //jika animasi aktif
47. if (simAktif == 1) index = index + 1;
48. hapusLayar();
49. //menampilkan teks
50. teks("Gerak dan Kecepatan", 0.5*(canvas.width), 40,
'bold 18pt Calibri', 'blue', 'center');
51. teks("Simulasi perbandingan pengaruh kecepatan dan
akselerasi antara 2 objek", 0.5*(canvas.width), 75,
"12pt Calibri", "#000", "center");
52.
53. grafik(graf);
54.
55. //slider
56. teks("Mobil Merah", 50, 370, "bold 15pt Calibri", "red",
"left");
57. teks("Posisi Awal = "+x1Init+ " m", 50, 400, "bold 13pt
Calibri", "black", "left");
58. slider(slider1);
59. teks("Kecepatan = "+v1+ " m/s", 50, 460, "bold 13pt
Calibri", "black", "left");
60. slider(slider2);
61. teks("Akselerasi = "+a1+ " m/s2", 50, 520, "bold 13pt
Calibri", "black", "left");
62. slider(slider3);
63.
64. teks("Mobil Biru", 450, 370, "bold 15pt Calibri",
"blue", "left");
65. teks("Posisi Awal = "+x2Init+ " m", 450, 400, "bold 13pt
Calibri", "black", "left");
66. slider(slider4);

```

```

67. teks("Kecepatan = "+v2+ " m/s", 450, 460, "bold 13pt
 Calibri", "black", "left");
68. slider(slider5);
69. teks("Akselerasi = "+a2+ " m/s2", 450, 520, "bold 13pt
 Calibri", "black", "left");
70. slider(slider6);
71.
72. //tombol control
73. tombol("Play", 150,300, 80, 30, "bold 11pt Calibri",
 "white", "black", "gray", "r");
74. tombol("Pause", 240,300, 80, 30, "bold 11pt Calibri",
 "white", "black", "gray", "r");
75. tombol("<< Step", 330,300, 80, 30, "bold 11pt Calibri",
 "white", "black", "gray", "r");
76. tombol("Step >>", 420,300, 80, 30, "bold 11pt Calibri",
 "white", "black", "gray", "r");
77. tombol("Reset", 510,300, 80, 30, "bold 11pt Calibri",
 "white", "black", "gray", "r");
78.
79. //menggambar jejak mobil merah pada diagrams
80. time = index/20.0;
81. var jumlahTitik = Math.round(0.5*time+0.5);
82. var vscale = graf.tileW/10;
83.
84. for (var i = 0; i < jumlahTitik; i++){
85. x1 = xBase + vscale*(x1Init +v1*i*2 + 0.5*a1*i*i*4);
86. lingkaran(x1, y1, radius/2, 1, "#000", "red");
87. }
88.
89. //menambahkan mobil merah ke layar
90. x1 = xBase + vscale*(x1Init +v1*time + 0.5*a1*
 time*time);
91. gambar(dataGambar.mobil_merah, x1, y1-12);
92.
93. //menggambar jejak mobil biru pada diagrams
94. for (i = 0; i < jumlahTitik; i++){
95. x2 = xBase + vscale*(x2Init +v2*i*2 + 0.5*a2*i*i*4);
96. lingkaran(x2, y2, radius/2, 1, "#000", "blue");
97. }
98.
99. //menambahkan mobil biru ke layar
100. x2 = xBase + vscale*(x2Init +v2*time + 0.5*a2*time*time);
101. gambar(dataGambar.mobil_biru, x2, y2-12);
102.
103. //menambahkan teks dan jarak
104. var timeLabel = 'Waktu (t) = ';
105. timeLabel = timeLabel + time.toFixed(2) + ' s';
106. teks(timeLabel, 50, 120, "bold 14pt Calibri", "#000",
 "left");
107. var xPos = (x1Init +v1*time + 0.5*a1*time*time);
108. var xPosLabel = 'd1 = ';
109. xPosLabel = xPosLabel + xPos.toFixed(2) + ' m';
110. teks(xPosLabel, 280, 370, "bold 14pt Calibri", "red",
 "left");

```

```

111. var xPos2 = (x2Init +v2*time + 0.5*a2*time*time);
112. xPosLabel = 'd2 = ';
113. xPosLabel = xPosLabel + xPos2.toFixed(2) + ' m';
114. teks(xPosLabel, 670, 370, "bold 14pt Calibri", "blue",
115. "left");
116.
117. function mouseDown(event){
118. canvas.onmousemove = mouseDrag;
119. }
120.
121. function mouseUp(event){
122. //prosedure mengecek tombol
123. var tombolAktif = cekTombol(event);
124. if (tombolAktif != ""){
125. if (tombolAktif == "Play"){
126. window.clearTimeout(timer);
127. simAktif = 1;
128. jalankanSimulasi();
129. }
130. if (tombolAktif == "Reset"){
131. reset();
132. }
133. if (tombolAktif == "Pause"){
134. window.clearTimeout(timer);
135. simAktif = 0;
136. }
137. if (tombolAktif == "<< Step"){
138. window.clearTimeout(timer);
139. index = index-5;
140. if (index < 0) index = 0;
141. time = index/20;
142. simAktif = 1;
143. setSimulasi();
144. }
145. if (tombolAktif == "Step >>"){
146. window.clearTimeout(timer);
147. if ((x1 < (xBase+graf.tileW*graf.dataW)) &&
148. (x2 < (xBase+graf.tileW*graf.dataW))){
149. index = index+5;
150. simAktif = 1;
151. setSimulasi();
152. }
153. }
154. //menetralisir drag
155. canvas.onmousemove = null;
156. }
157.
158. function mouseDrag(event){
159. //prosedur mengecek slider
160. var sliderAktif = cekSlider(event);

```

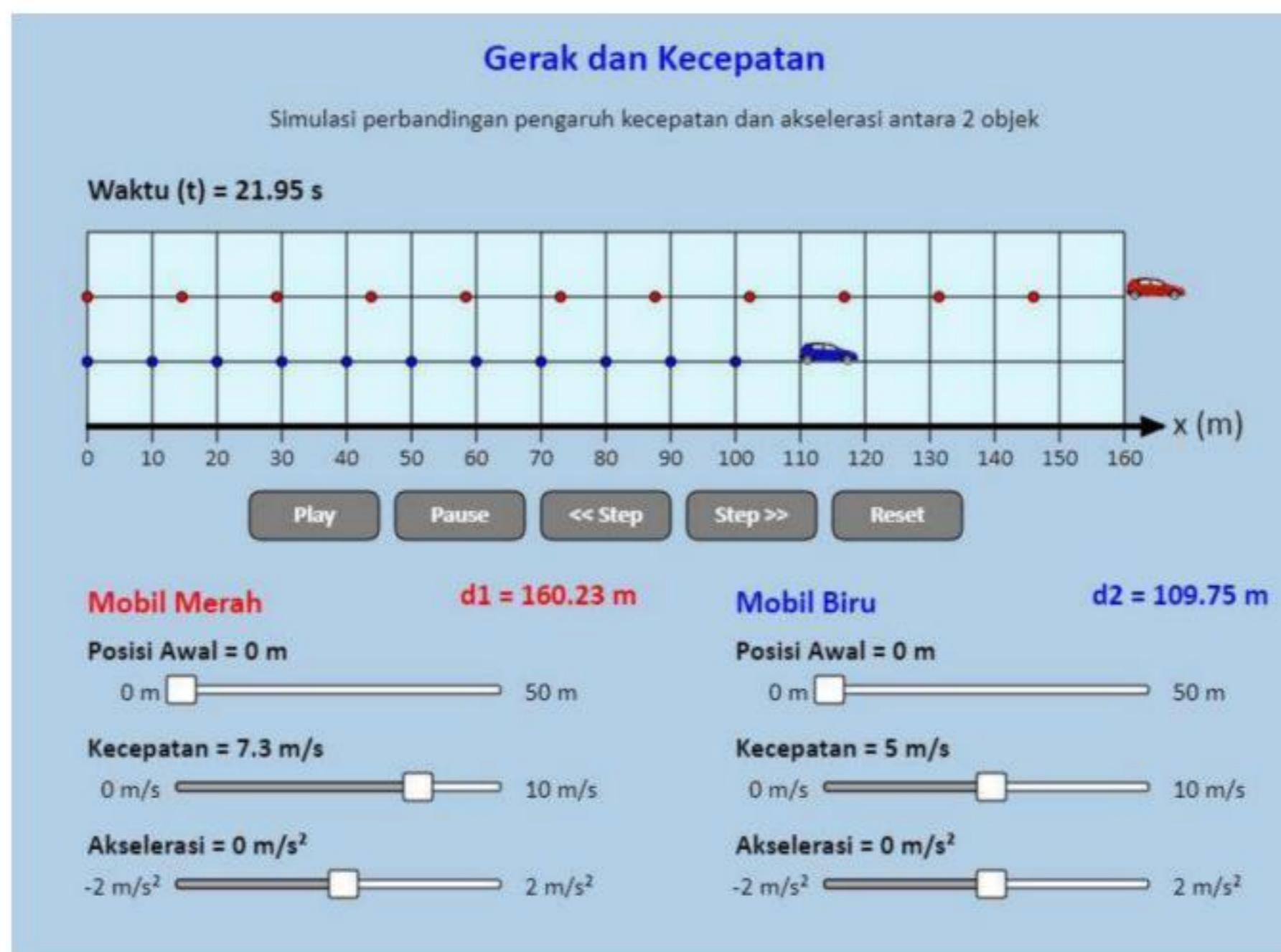
```

161. if (sliderAktif != null){
162. if (sliderAktif.nama == "posisi1") x1Init =
163. Number(sliderAktif.vals);
164. if (sliderAktif.nama == "kecepatan1") v1 =
165. Number(sliderAktif.vals);
166. if (sliderAktif.nama == "acc1") a1 =
167. Number(sliderAktif.vals);
168. //netralkan posisi objek ke awal
169. index = 0;
170. simAktif = 0;
171. setSimulasi();
172. }
173. }
174.
175. function jalankanSimulasi() {
176. setSimulasi();
177. if (simAktif == 1) {
178. timer = window.setTimeout(jalankanSimulasi, 20);
179. }
180. }
181.
182. function reset(){
183. window.clearTimeout(timer);
184. simAktif = 0;
185. index = 0;
186. time = 0.0;
187. x1Init = 0;
188. v1 = 5;
189. a1 = 0;
190. x2Init = 0;
191. v2 = 5;
192. a2 = 0;
193. slider1.vals = 0;
194. slider2.vals = 5;
195. slider3.vals = 0;
196. slider4.vals = 0;
197. slider5.vals = 5;
198. slider6.vals = 0;
199. jalankanSimulasi();
200. }

```

3. Simpan dengan nama **simulasi-9.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-9.

Jalankan file **tutorial-9.html**, dengan cara membukanya di *internet browser*. Maka kode di atas akan menghasilkan simulasi gerak dan kecepatan antara 2 mobil sebagai berikut :



Gambar 9.1 Hasil tutorial gerak dan kecepatan

#### d. Penjelasan Program

Seperti halnya pada tutorial sebelumnya, diperlukan beberapa variabel untuk mengatur elemen dari simulasi seperti grafik untuk *time ticker*, beberapa *slider* untuk mengatur variabel dinamis yang dalam hal ini adalah posisi awal mobil, kecepatan dan percepatannya. Selanjutnya dibutuhkan 2 gambar *bitmap*, yaitu mobil merah dan biru, yang selanjutnya dibuka dengan fungsi *preload*.

Dalam simulasi ini terdapat fitur baru yang diperkenalkan yaitu seperangkat tombol untuk mengatur jalannya simulasi, mulai dari tombol *play* hingga tombol *reset*. Dalam simulasi, tombol sangat bermanfaat untuk mengetahui secara detail jalannya simulasi, sebagai contoh kita dapat melihat perubahan setiap langkah pergerakan objek dengan menekan tombol maju atau mundur. Dengan tombol *reset* dan *play*, kita dapat memberikan masukan variabel yang berbeda-beda, memainkannya kembali, sehingga kita dapat mengetahui hubungan antar variabel terhadap gerakan objek.

Kunci untuk mengatur jalannya simulasi sebagaimana dijelaskan pada paragraf di atas adalah variabel `simAktif`, dan variabel `index`. Variabel `simAktif` digunakan untuk mengatur status simulasi, ketika aktif (bernilai 1) maka setiap langkah akan menambahkan variabel `index` sebanyak 1 satuan. Variabel `index` ini nantinya berperan sebagai media untuk menyimpan satuan waktu dan digunakan untuk menghitung gerakan objek berdasarkan satuan waktu tersebut (lihat baris 80). Variabel waktu dihitung berdasarkan variabel `index`.

```
80. time = index/20.0;
```

Sebagai catatan simulasi pada tutorial-9 ini dijalankan menggunakan fungsi `setTimeOut` pada baris 178.

```
178. timer = window.setTimeout(jalankanSimulasi, 20);
```

Fungsi `setTimeout` tersebut akan mengeksekusi fungsi `jalankanSimulasi` sebanyak 1 kali setelah menunggu 20 mili detik. Inilah alasan kenapa pada baris 80 variabel `time` bernilai `index` dibagi dengan angka 20, agar menghasilkan satuan waktu standar yaitu detik (s).

Variabel `vscale` (baris 82) digunakan untuk melakukan penskalaan pada *timeticker* agar disesuaikan dengan lebar tiap satuan pada grafik. Nilai `vscale` merupakan lebar grafik dibagi 10, dikarenakan penskalaan grafik (`graf.skalaX`) menggunakan kelipatan 10. Selanjutnya untuk menggambar *ticker* pada grafik digunakan operasi berulang `for` untuk menggambar beberapa titik lingkaran.

Untuk menggerakkan mobil dilakukan perhitungan jarak yaitu

$$s = v \cdot t$$

Namun karena pada simulasi ini juga dimungkinkan terdapat akselerasi atau percepatan, maka untuk menentukan jarak digunakan rumus

$$s = v \cdot t + (a \cdot t \cdot t)/2$$

Rumus ini dijabarkan pada baris 90, dimana jarak (`x1`), harus ditambahkan terlebih dahulu dengan posisi awal mobil (titik 0 pada grafik) yaitu `xBase`, dan perlu ditambahkan dengan `x1Init`, yaitu variabel yang menentukan posisi awal dari mobil.

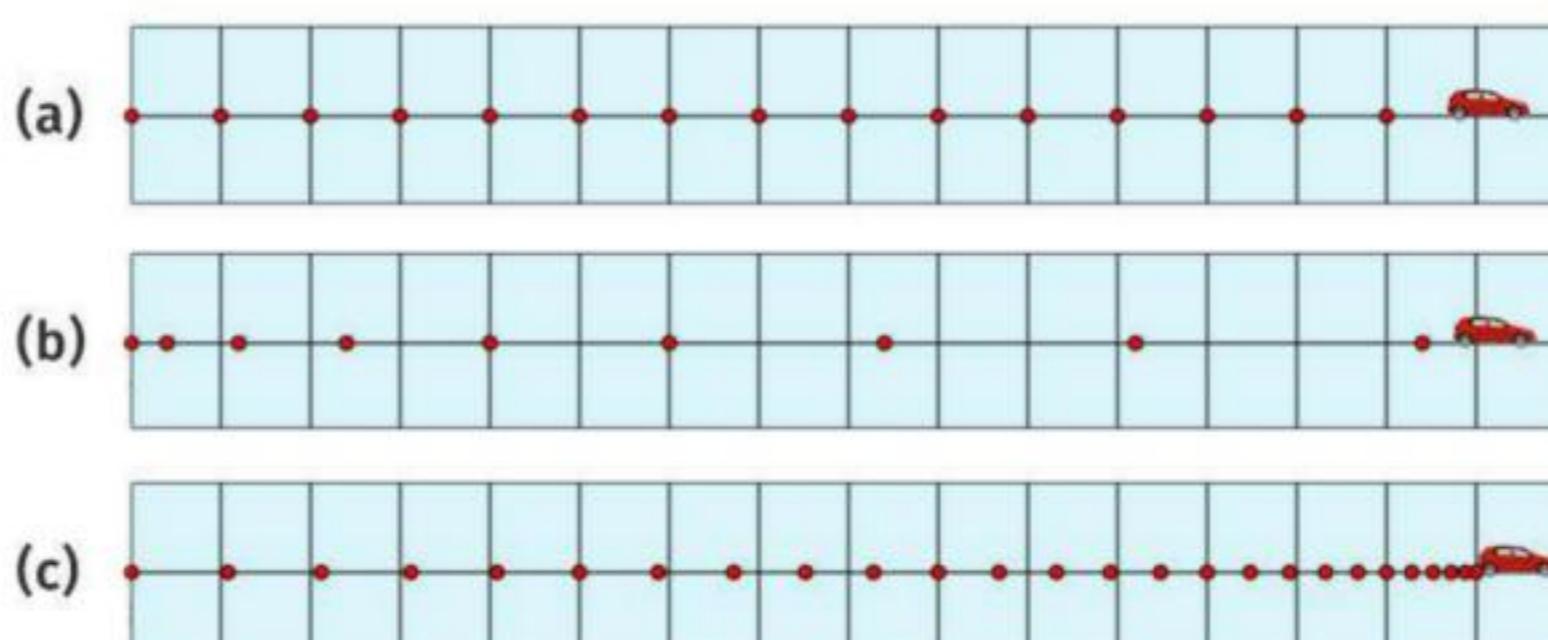
```
90 x1 = xBase + vscale*(x1Init + v1*time + 0.5*a1*
 time*time);
```

Kode tersebut di atas merupakan format kode dari formula perhitungan jarak pada Gerak Lurus Berubah Beraturan :

$$s = v_0 \cdot t + (a \cdot t \cdot t) / 2$$

Variabel `x1` yang merupakan jarak yang ditempuh oleh mobil merah, perlu dikalikan dengan `vscale` yang merupakan variabel dalam transformasi linear jarak ke dalam koordinat layar. Hal ini bertujuan untuk menyesuaikan posisi jarak yang sesungguhnya ke dalam format grafik. Tanpa `vscale` seluruh perhitungan menggunakan ukuran 1 satuan *pixel*, sehingga 1 *pixel* dianggap sebesar 1 meter dan simulasi akan berjalan lebih lambat serta *ticker* tidak akan digambarkan dengan tepat pada grafik.

Dari simulasi tersebut, diharapkan peserta didik dapat mengetahui jenis-jenis gerak lurus, baik itu gerak lurus beraturan (tanpa penambahan variabel akselerasi/percepatan), atau gerak lurus berubah beraturan (dengan penambahan akselerasi/percepatan). Simulasi juga dapat menunjukkan perubahan kecepatan dan percepatan seiring waktu, yang ditunjukan pada grafik dalam bentuk titik-titik, dimana pada percobaan lab yang sesungguhnya diperlukan *time ticker* dan pita kertas.



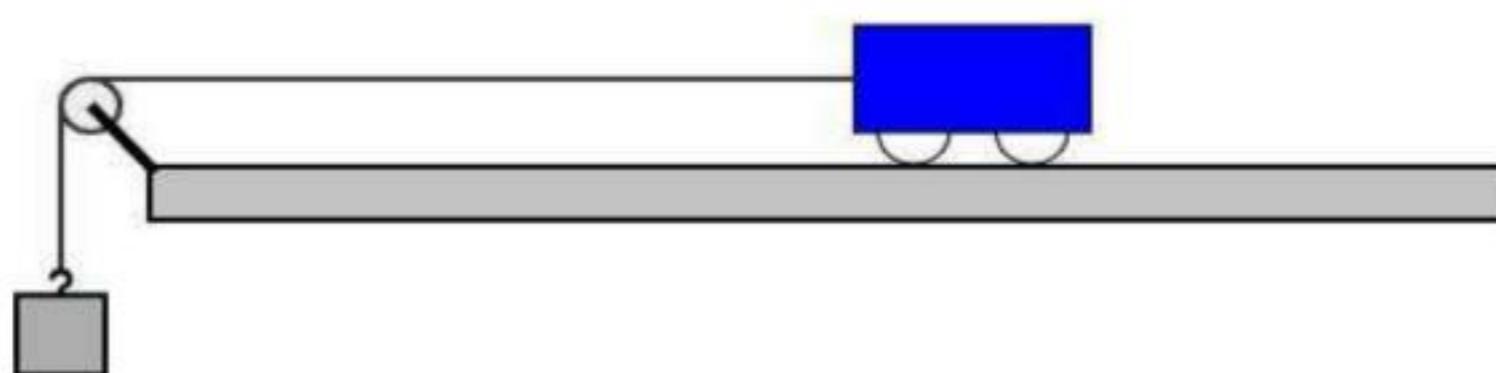
Gambar 9.2 Hasil Simulasi

(a) GLBB (b) GLBB dipercepat (c) GLBB diperlambat

## B. Tutorial 10 – Simulasi Gerak Lurus Berubah Beraturan

Pada tutorial berikut, kita akan membuat sebuah simulasi praktikum Gerak Lurus Berubah Beraturan (GLBB) yang melibatkan kereta luncur, beban massa dan pewaktu ketik (*time ticker*). Praktikum ini menjadi salah satu praktikum standar pada bidang fisika. Dengan mensimulasikan praktikum, peserta didik dapat melaksanakan praktikum secara berulang, memahami variabel yang terlibat dan menyimpulkan praktikum. Simulasi seperti ini bermanfaat sebagai salah satu metode untuk menyiapkan peserta didik pada praktikum yang sesungguhnya di laboratorium.

Model simulasi yang akan dibuat pada tutorial selanjutnya adalah sebagai berikut :



Gambar 9.3 Model simulasi GLBB dengan beban massa

Pada model ini beberapa variabel yang terlibat di antaranya massa troli ( $m_1$ ), massa beban ( $m_2$ ), tegangan tali ( $T$ ), waktu ( $t$ ), jarak gerak troli ( $s$ ), gaya gravitasi ( $g$ ) dan percepatan troli ( $a$ ). Pada simulasi tersebut kita akan mencari percepatan troli yang diperoleh dari perhitungan tegangan tali (*tension*), sehingga persamaan yang digunakan adalah sebagai berikut :

$$T = m_2 \cdot g \quad (\text{ketika troli dalam keadaan diam})$$

$$T = m_1 \cdot m_2 \cdot g / (m_1 + m_2) \quad (\text{ketika troli bergerak})$$

Yang selanjutnya dapat digunakan untuk mendapatkan percepatan troli, dengan persamaan sebagai berikut :

$$a = T / m_1$$

Dari percepatan tersebut, dapat dihitung GLBB dari troli dan beban dengan persamaan sebagai berikut :

$$s = v_0 \cdot t + (a \cdot t \cdot t) / 2$$

Untuk mengaplikasikan persamaan-persamaan tersebut ke dalam simulasi, ikutilah tutorial berikut :

**a. File HTML**

1. Seperti pada projek tutorial sebelumnya, buatlah sebuah *folder* khusus dengan nama **tutorial-10**. *Copy file tutorial-9.html* dan *paste* ke dalam *folder*.
2. *Rename* nama *file* menjadi **tutorial-10.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
12. <script src="simulasi-10.js"></script>
```

3. Simpan kembali *file* HTML.

**b. File library**

1. *Copy paste* *file* **simulasiDasar.js** ke dalam *folder* **tutorial-10**.

**c. File simulasi-10.js**

1. Pada Notepad++, buatlah sebuah *file* baru
2. Ketikkan kode berikut :

```
1. aturCanvas();
2. setJudul("Gerak Lurus Berubah Beraturan");
3. hapusLayar("#b3cfe5");
4.
5. //listener untuk membaca event mouse
6. canvas.onmousedown = mouseDown;
7. canvas.onmouseup = mouseUp;
8.
9. var gridData = {startX:110, startY:210, skalaGrid:20,
10. total:10, warnaGaris:"#000", warnaLabel:"#000",
11. fontLabel:"12pt Calibri", labelY:5, label:["0 cm", "50 cm",
12. "100 cm"] }
13.
14. var slider1 = {tipe:"H", nama:"waktu", x:280,y:460, p:200,
15. minS:0, maxS:10, valS:0, desimal:1, label:"s"}
16. var slider2 = {tipe:"H", nama:"massa", x:280,y:520, p:200,
17. minS:20, maxS:200, valS:100, desimal:1, label:"g"}
18.
19. var index = -1;
20. var g = 10.0;
21. var xBase = 140;
22. var yBase = 100;
23. var yStart = yBase+120;
24. var radius = 12;
25. var x1Init = 300;
26. var x1 = x1Init;
27. var v1 = 0.0;
28. var a1 = 400.0;
29. var m1 = 0.2;
```

```

25. var m2 = 0.05;
26. var tension = m2*g;
27. var y1 = yBase + 20;
28. var y2Init = yBase + 80;
29. var y2 = y2Init;
30. var maxTime = Math.sqrt(400.0/a1);
31. var maxIndex = Math.round(200.0*maxTime);
32. maxTime = maxIndex/200.0;
33. var time = 0.0;
34. var timer;
35. var simAktif = 1;
36.
37. function setSimulasi(){
38. if ((index >= 200) || (time >= maxTime) ||
39. (y2 >= (y2Init+200))) simAktif = 0;
40. //jika animasi aktif
41. if (simAktif == 1) index = index + 1;
42. //hapus layar
43. hapusLayar();
44. //menampilkan teks
45. teks("Gerak Lurus Berubah Beraturan", 0.5*(canvas.width),
46. 40, 'bold 18pt Calibri', 'blue', 'center');
47. teks("Simulasi gerak lurus berubah beraturan pada kereta
48. yang ditarik beban", 0.5*(canvas.width), 75, "12pt
49. Calibri", "#000", "center");
50.
51. teks("Waktu = "+(time*10).toFixed(2)+ " s", 260, 440,
52. "bold 13pt Calibri", "black", "left");
53. slider(slider1);
54. teks("Massa = "+(m2*1000)+ " g", 260, 500, "bold 13pt
55. Calibri", "black", "left");
56. slider(slider2);
57.
58. //tombol control
59. tombol("Play", 150,560, 80, 30, "bold 11pt Calibri",
60. "white", "black", "gray", "r");
61. tombol("Pause", 240,560, 80, 30, "bold 11pt Calibri",
62. "white", "black", "gray", "r");
63. tombol("<< Step", 330,560, 80, 30, "bold 11pt Calibri",
64. "white", "black", "gray", "r");
65. tombol("Step >>", 420,560, 80, 30, "bold 11pt Calibri",
66. "white", "black", "gray", "r");
67. tombol("Reset", 510,560, 80, 30, "bold 11pt Calibri",
68. "white", "black", "gray", "r");
69.
70. time = index/200.0;
71. if (index > 0.5){
72. tension = m1*m2*g/(m1+m2);
73. }else {
74. tension = m2*g;
75. }
76.
77. a1 = 200*tension/m1;
78.
79. }
80.
```

```

68. x1 = xBase + x1Init - v1*time - 0.5*a1*time*time;
69. y2 = y2Init + v1*time + 0.5*a1*time*time;
70.
71. //ubah slider waktu
72. slider1.valS = time*10;
73.
74. // track
75. kotak(160, yBase+73, 460, 20, 2, "#000", '#c3c3c3');
76.
77. //kereta
78. lingkaran(x1+20, y1+40, radius);
79. lingkaran(x1+60, y1+40, radius);
80. kotak(x1,y1, 80, 40, 1, "#000", "blue");
81.
82. // katrol
83. lingkaran(xBase, y1+30, radius-2, 2, "#333", "#f0f0f0");
84. garis(xBase, y1+30, 162, yBase+75, 4, "#000");
85.
86. //tali
87. garis(xBase, y1+20, x1, y1+20,2, "#333");
88. garis(xBase-10, y1+30, xBase-10, y2-9, 2, "#333");
89. pengait(xBase-10, y2-9, 9, 2);
90.
91. // beban
92. kotak(xBase-25, y2, 30,30,2, "#000", "#adadad");
93.
94. //grid
95. gridV(gridData);
96. }
97.
98. function mouseDown(event){
99. canvas.onmousemove = mouseDrag;
100. }
101.
102. function mouseUp(event){
103. //prosedure mengecek tombol
104. var tombolAktif = cekTombol(event);
105. if (tombolAktif != ""){
106. if (tombolAktif == "Play"){
107. if ((index >= 200) || (time >= maxTime) ||
108. (y2 >= (y2Init+200))) reset();
109. window.clearTimeout(timer);
110. simAktif = 1;
111. jalankanSimulasi();
112. if (tombolAktif == "Reset")reset();
113. if (tombolAktif == "Pause"){
114. window.clearTimeout(timer);
115. simAktif = 0;
116. }
117. if (tombolAktif == "<< Step"){


```

```

118. window.clearTimeout(timer);
119. index = index-5;
120. if (index < 0) index = 0;
121. time = index/200;
122. xPos = xBase;
123. y2 = y2Init;
124. simAktif = 1;
125. setSimulasi();
126. }
127. if (tombolAktif == "Step >>"){
128. window.clearTimeout(timer);
129. if (((index < 200) && (time < maxTime) && (y2 <
130. (y2Init+200)))){
131. index = index+5;
132. }
133. simAktif = 1;
134. setSimulasi();
135. }
136. //menetralisir drag
137. canvas.onmousemove = null;
138. }
139.
140. function mouseDrag(event){
141. //prosedur mengecek slider
142. var sliderAktif = cekSlider(event);
143. if (sliderAktif != null){
144. if (sliderAktif.nama == "waktu") {
145. time = Number(sliderAktif.vals)/10;
146. index = 200*time;
147. }
148. if (sliderAktif.nama == "massa") {
149. var mas = Number(sliderAktif.vals);
150. index = 0;
151. m2 = mas/1000.0;
152. }
153. //atur ulang variabel
154. tension = m1*m2*g/ (m1+m2);
155. a1 = 200*tension/m1;
156. maxTime = Math.sqrt(400.0/a1);
157. maxIndex = Math.round(200.0*maxTime);
158. maxTime = maxIndex/200.0;
159. if (index > maxIndex) {
160. index = maxIndex;
161. }
162. y2 = y2Init;
163. simAktif = 1;
164. setSimulasi();
165. }
166. }
167.

```

```

168. function jalankanSimulasi() {
169. setSimulasi();
170. if (simAktif == 1) {
171. timer = window.setTimeout(jalankanSimulasi, 20);
172. }
173. }
174.
175. function reset(){
176. window.clearTimeout(timer);
177. simAktif = 0;
178. index = -1;
179. time = 0.0;
180. xPos = xBase;
181. y2 = y2Init;
182. slider1.vals = 0;
183. slider2.vals = 50;
184. jalankanSimulasi();
185. }
186.
187. function jalankanSimulasi() {
188. setSimulasi();
189. if (simAktif == 1) {
190. timer = window.setTimeout(jalankanSimulasi, 10);
191. }
192. }
193.
194. setSimulasi();

```

3. Simpan dengan nama **simulasi-10.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-10.

Jalankan *file* **tutorial-10.html**, dengan cara membukanya di internet *browser*. Maka kode di atas menghasilkan simulasi gerak kereta (troli) dan beban berikut:



Gambar 9.4 Simulasi hubungan kecepatan dan massa

#### d. Penjelasan Program

Seperti halnya pada tutorial sebelumnya, kunci dari simulasi adalah variabel `index` dan perhitungan rumus fisika terkait gerak, kecepatan dan massa benda (lihat baris 59 – 69). Pada bagian awal ditentukan besaran gravitasi, massa beban, serta 2 buah *slider* untuk mengatur variabel waktu dan massa secara dinamis. Selanjutnya elemen visual dari simulasi tersebut digambarkan secara berulang di layar, seiring jalankannya simulasi.

Pada baris 59, variabel time memiliki nilai variabel `index` di bagi dengan 200.

```
59. time = index/200;
```

Demikian halnya dengan variabel percepatan (`a1`), yang seharusnya bernilai `tension/m1` juga dikalikan dengan angka 200.

```
66. a1 = 200*tension/m1;
```

Angka 200 tersebut diperoleh dari jarak yang ditempuh oleh troli dalam satuan *pixel*. Pada praktikum yang sesungguhnya, jarak yang ditempuh oleh troli adalah sejauh 1 meter, yang dipresentasikan dalam ukuran layar sejauh 200 *pixel*. Untuk mendapatkan posisi yang tepat pada layar, maka variabel-variabel yang ada harus ditransformasikan secara linear ke dalam sistem koordinat layar.

## 9.2 Gerak Parabola

Gerak parabola merupakan perpaduan gerak lurus dengan kecepatan konstan dan percepatan konstan yang menghasilkan lintasan parabola. Gerak parabola terjadi jika suatu objek diluncurkan pada medan gravitasi bumi dengan kecepatan tertentu dan membentuk sudut theta terhadap arah horisontal (sudut elevasi). Objek mendapat percepatan searah dengan medan gravitasi bumi yaitu ke arah pusat massa bumi. Akibatnya pada arah vertikal objek mengalami GLBB diperlambat/dipercepat. Contoh dari gerak parabola adalah gerakan peluru yang ditembakkan dengan sudut tertentu, gerakan lompat jauh, gerakan bola ketika ditendang ke depan, dan sebagainya. Persamaan yang digunakan untuk menghitung gerak parabola adalah sebagai berikut :

$$s_x = v \cdot \cos(\theta) \cdot t$$

$$s_y = v \cdot \sin(\theta) \cdot t + (a \cdot t \cdot t) / 2$$

Selanjutnya perhatikan tutorial berikut, untuk mensimulasikan gerak parabola:

#### A. Tutorial 11 – Gerak Parabola

##### a. File HTML

1. Seperti pada projek tutorial sebelumnya, buatlah sebuah *folder* khusus dengan nama **tutorial-11**. *Copy file tutorial-10.html* dan *paste* ke dalam *folder*.
2. *Rename* nama *file* menjadi **tutorial-10.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
12. <script src="simulasi-11.js"></script>
```

3. Simpan kembali *file* HTML.

##### b. File library

1. *Copy paste* *file* **simulasiDasar.js** ke dalam *folder* **tutorial-11**.

##### c. File simulasi-11.js

1. Pada Notepad++, buatlah sebuah *file* baru
2. Ketikkan kode berikut :

```
1. aturCanvas();
2. setJudul("Gerak Parabola");
3. hapusLayar("#b3cfe5");
4.
5. //listener untuk membaca event mouse
6. canvas.onmousedown = mouseDown;
7. canvas.onmouseup = mouseUp;
8.
9. var graf = {startX:180, startY:150, dataW:10, dataH:4,
tileW:40, skalaX:10, skalaY:10, desimalX:0, desimalY:0,
offsetX:0, offsetY:2, xLabel:'x (m)', yLabel:'y (m)',
fontLabel:'12pt Calibri', warnaBG:'#daf6fb',
warnaGaris:'#000', warnaLabel:'#000'}
10.
11. var slider1 = {tipe:"H", nama:"Kecepatan", x:280,y:460,
p:200, minS:10, maxS:30, vals:20, desimal:1, label:"m/s"}
12. var slider2 = {tipe:"H", nama:"Sudut", x:280,y:520, p:200,
minS:0, maxS:90, vals:45, desimal:0, label:"°"}
13.
14. var index = -1;
15. var xBase = 180;
16. var yBase = 150;
17. var yStart = yBase+160;
18. var warnaGaris = 'green';
19. var radius = 3;
20. var a = -10.0;
```

```

21. var vInit = 20.0;
22. var angle = 45.0;
23. var theta = angle*Math.PI/180.0;
24. var vInitx = vInit*Math.cos(theta);
25. var vInity = vInit*Math.sin(theta);
26. var maxTime = -2.0*vInity/a;
27. var time = 0.0;
28. var range = vInitx*maxTime;
29. var maxHeight = -vInity*vInity/(2.0*a);
30. var timer;
31. var simAktif = 1;
32.
33. function setSimulasi(){
34. if ((yPos < 0) || (time == maxTime)) simAktif = 0;
35. //jika animasi aktif
36. if (simAktif == 1) index = index + 1;
37. //hapus layar
38. hapusLayar();
39.
40. //menampilkan teks
41. teks("Gerak Parabola", 0.5*(canvas.width), 40, 'bold 18pt
Calibri', 'blue', 'center');
42. teks("Simulasi gerak peluru yang diluncurkan dengan
kecepatan awal dan sudut tertentu", 0.5*(canvas.width),
75, "12pt Calibri", "#000");
43.
44. grafik(graf);
45.
46. teks("Kecepatan awal = "+vInit+ " m/s", 260, 440, "bold
13pt
Calibri", "black", "left");
47. slider(slider1);
48. teks("Sudut = "+angle+ " °", 260, 500, "bold 13pt
Calibri", "black", "left");
49. slider(slider2);
50.
51. //tombol control
52. tombol("Play", 150,560, 80, 30, "bold 11pt Calibri",
"white", "black", "gray", "r");
53. tombol("Pause", 240,560, 80, 30, "bold 11pt Calibri",
"white", "black", "gray", "r");
54. tombol("<< Step", 330,560, 80, 30, "bold 11pt Calibri",
"white", "black", "gray", "r");
55. tombol("Step >>", 420,560, 80, 30, "bold 11pt Calibri",
"white", "black", "gray", "r");
56. tombol("Reset", 510,560, 80, 30, "bold 11pt Calibri",
"white", "black", "gray", "r");
57.
58. //menggambar grafis parabola
59. var vscale = 4;
60. konten.strokeStyle = warnaGaris;
61. konten.lineWidth = 3;
62. konten.beginPath();
63. konten.moveTo(xBase, yStart);

```

```

64. for (var ival = 1; ival <=index; ival++) {
65. var posX = xBase+vscale*vInitx*ival/20.0;
66. var posY = yStart-vscole*vInity*(ival/20.0)-
67. vscole*0.5*a*(ival/20.0)*(ival/20.0);
68. konten.lineTo(posX, posY);
69. }
70. konten.stroke();
71. //menggambar peluru lingkaran
72. time = index/20.0;
73. posX = xBase+vscole*vInitx*time;
74. posY = yStart-vscole*vInity*time-vscole*0.5*a*time*time;
75. lingkaran(posX, posY, radius, 1, "#000", "red");
76.
77. var yPos = vInity*time+0.5*a*time*time;
78. if (yPos < 0.0){
79. time = maxTime;
80. yPos = 0.0;
81. }
82.
83. //menambahkan teks
84. var timeLabel = 't = ';
85. timeLabel = timeLabel + time.toFixed(2) + ' s';
86. teks(timeLabel, 130, 380, "bold 14pt Calibri", "#000",
87. "left");
88. var xPos = vInitx*time;
89. var xPosLabel = 'x = ';
90. xPosLabel = xPosLabel + xPos.toFixed(2) + ' m';
91. teks(xPosLabel, 230, 380, "bold 14pt Calibri", "#000",
92. "left");
93. var yPosLabel = 'y = ';
94. yPosLabel = yPosLabel + yPos.toFixed(2) + ' m';
95. teks(yPosLabel, 330, 380, "bold 14pt Calibri", "#000",
96. "left");
97. teks("g = 10 m/s2", 430, 380, "bold 14pt Calibri",
98. "#000", "left");
99. var maxHLabel = 'tinggi max = ';
100. maxHLabel = maxHLabel + maxHeight.toFixed(2) + ' m';
101. teks(maxHLabel, 550, 380, "bold 14pt Calibri", "#000",
102. "left");
103.
104. function mouseDown(event){
105. canvas.onmousemove = mouseDrag;
106. }
107.
108. function mouseUp(event){
109. //prosedure mengecek tombol
110. var tombolAktif = cekTombol(event);

```

```

111. if (tombolAktif != "") {
112. if (tombolAktif == "Play") {
113. window.clearTimeout(timer);
114. simAktif = 1;
115. index = 0;
116. yPos = 0;
117. jalankanSimulasi();
118. }
119. if (tombolAktif == "Reset") reset();
120. if (tombolAktif == "Pause") {
121. window.clearTimeout(timer);
122. simAktif = 0;
123. }
124. if (tombolAktif == "<< Step") {
125. window.clearTimeout(timer);
126. index = index-5;
127. if (index < -1) index = -1;
128. time = index/20;
129. xPos = xBase;
130. simAktif = 1;
131. setSimulasi();
132. }
133. if (tombolAktif == "Step >>") {
134. window.clearTimeout(timer);
135. if ((yPos > 0) || (time < maxTime)) index = index+5;
136. simAktif = 1;
137. setSimulasi();
138. }
139. }
140. //menetralisir drag
141. canvas.onmousemove = null;
142. }
143.
144. function mouseDrag(event){
145. //prosedur mengecek slider
146. var sliderAktif = cekSlider(event);
147. if (sliderAktif != null){
148. if (sliderAktif.nama == "Kecepatan") {
149. index = 0;
150. vInit = Number(sliderAktif.valS);
151. vInitx = vInit*Math.cos(theta);
152. vInity = vInit*Math.sin(theta);
153. maxTime = -2.0*vInity/a;
154. range = vInitx*maxTime;
155. maxHeight = -vInity*vInity/(2.0*a);
156. }
157. if (sliderAktif.nama == "Sudut") {
158. index = 0;
159. angle = Number(sliderAktif.valS);
160. theta = angle*Math.PI/180.0;
161. vInitx = vInit*Math.cos(theta);

```

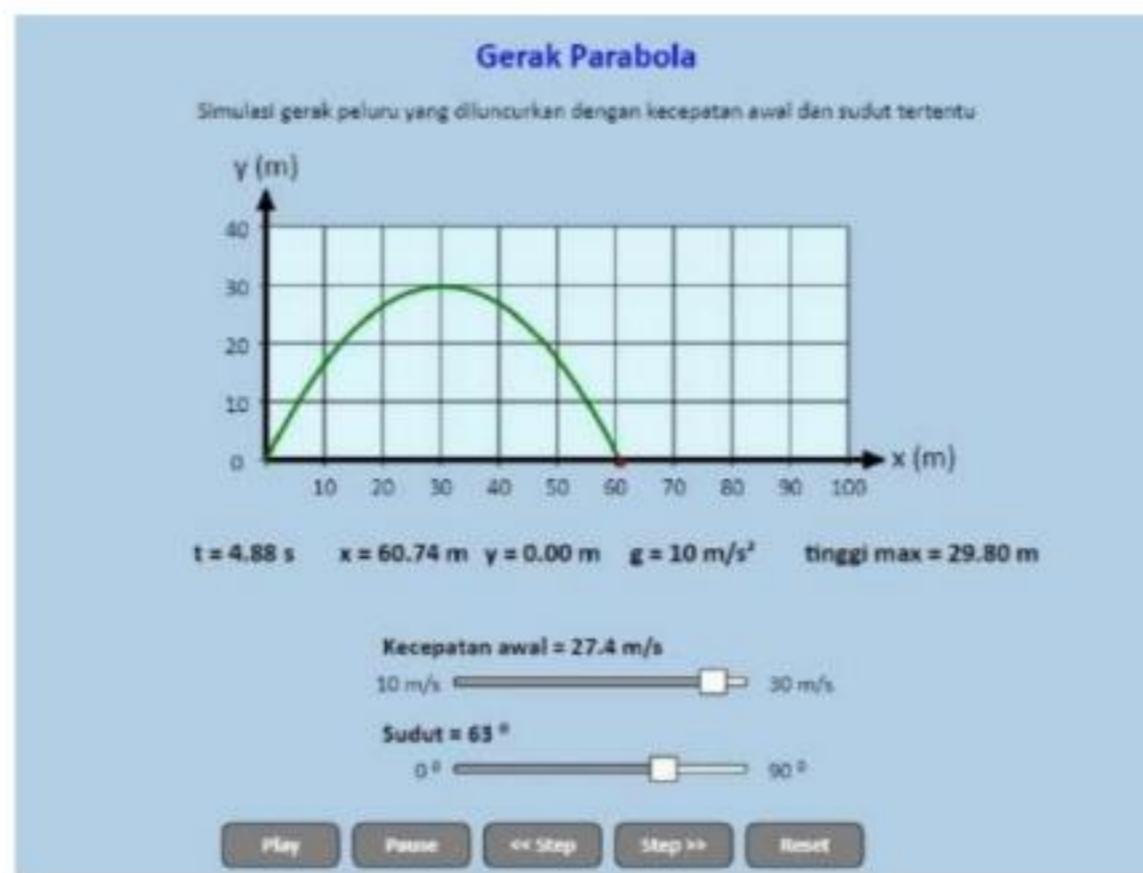
```

162. vInity = vInit*Math.sin(theta);
163. maxTime = -2.0*vInity/a;
164. range = vInitx*maxTime;
165. maxHeight = -vInity*vInity/(2.0*a);
166. }
167. simAktif = 0;
168. setSimulasi();
169. }
170. }
171.
172. function jalankanSimulasi() {
173. setSimulasi();
174. if (simAktif == 1) {
175. timer = window.setTimeout(jalankanSimulasi, 20);
176. }
177. }
178.
179. function reset(){
180. window.clearTimeout(timer);
181. simAktif = 0;
182. index = 0;
183. time = 0.0;
184. xPos = xBase;
185. jalankanSimulasi();
186. }
187.
188. setSimulasi();

```

3. Simpan dengan nama **simulasi-11.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-11.

Jalankan *file* **tutorial-11.html**, dengan cara membukanya di *internet browser*. Maka kode di atas akan menghasilkan simulasi gerak parabola sebagai berikut :



Gambar 9.5 Hasil simulasi gerak parabola

#### d. Penjelasan Program

Seperti halnya pada tutorial sebelumnya, simulasi diatur oleh variabel index.

Perhitungan rumus fisika gerak parabola adalah sebagai berikut :

1. Tentukan kecepatan pada sumbu X (baris 24, dan ketika variabel diubah, maka diatur ulang pada baris 151 dan 161)

```
vInitx = vInit*Math.cos(theta);
```

2. Tentukan kecepatan pada sumbu Y (baris 25, dan ketika variabel diubah, maka diatur ulang pada baris 152 dan 162)

```
vInity = vInit*Math.sin(theta);
```

3. Posisi peluru ketika bergerak disumbu X merupakan gerak lurus beraturan, sehingga digunakan rumus GLB (baris 73). Variabel vscale merupakan variabel untuk mengatur skala untuk perhitungan posisi pada grafik (lihat penjelasan tutorial sebelumnya)

```
posX = xBase+vscale*vInitx*time;
```

4. Posisi peluru ketika bergerak disumbu Y merupakan gerak lurus berubah beraturan, karena terpengaruh gravitasi, sehingga digunakan rumus GLBB (baris 73). Variabel vscale merupakan variabel untuk mengatur skala untuk perhitungan posisi pada grafik (lihat penjelasan tutorial sebelumnya)

```
posY = yStart-vsacle*vInity*time-vsacle*0.5*a*time*time;
```

Note : perhitungan di atas kordinat Y justru dikurangi bukan ditambah, sementara pada perhitungan rumus GLBB ketinggian (kordinat Y) ditambah. Hal ini dikarenakan adanya perbedaan sistem kordinat antara sistem kordinat layar komputer dengan kordinat matematika standar. Pada kordinat layar, untuk menggerakkan objek ke atas, kordinat Y justru harus dikurangi.

### 9.3 Gerak Melingkar

Gerak melingkar merupakan gerak suatu benda yang membentuk lintasan berupa lingkaran mengelilingi suatu titik tetap. Agar suatu benda dapat bergerak melingkar ia membutuhkan adanya gaya sentripetal, yaitu gaya yang selalu membelokkan-nya menuju pusat lintasan lingkaran. Suatu gerak melingkar beraturan dapat dikatakan sebagai suatu gerak dipercepat beraturan, mengingat perlu adanya suatu percepatan yang besarnya tetap dengan arah yang berubah, yang selalu mengubah arah gerak benda agar menempuh lintasan berbentuk lingkaran. Persamaan yang digunakan dalam gerak melingkar adalah sebagai berikut :

$$v = r \cdot w$$

dimana  $v$  = kecepatan,  $r$  = jari-jari lingkaran dan  $w$  = kecepatan sudut maka untuk menemukan jarak perpindahan pada sumbu x dan y digunakan persamaan sebagai berikut :

$$s_x = r \cdot \cos(w \cdot t)$$

$$s_y = r \cdot \sin(w \cdot t)$$

Perhatikan tutorial berikut, untuk mensimulasikan gerak melingkar:

#### B. Tutorial 12 – Gerak Melingkar

##### a. File HTML

1. Seperti pada projek tutorial sebelumnya, buatlah sebuah *folder* khusus dengan nama **tutorial-12**. *Copy file tutorial-11.html* dan *paste* ke dalam *folder*.
2. *Rename* nama *file* menjadi **tutorial-12.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
12. <script src="simulasi-12.js"></script>
```

3. Simpan kembali *file* HTML.

##### b. File library

1. *Copy paste* *file* **simulasiDasar.js** ke dalam *folder* **tutorial-12**.

##### c. File simulasi-12.js

1. Pada Notepad++, buatlah sebuah *file* baru
2. Ketikkan kode berikut :

```

1. aturCanvas();
2. setJudul("Gerak Melingkar");
3. hapusLayar("#b3cfe5");
4.
5. //listener untuk membaca event mouse
6. canvas.onmousedown = mouseDown;
7. canvas.onmouseup = mouseUp;
8.
9. var index = -1;
10. var yBase = 280;
11. var xSupport = 250;
12. var ySupport = yBase;
13. var xBase = xSupport-180;
14. var xGraph = xSupport+270;
15. var yGraph = ySupport-160;
16. var time = 0.0;
17. var length = 1.2;
18. var rodLength = 100;
19. var omega = 2.0;
20. var radius = 6;
21. var xPos;
22. var yPos;
23. var timer;
24. var simAktif = 1;
25.
26. var slider1 = {tipe:"H", nama:"omega", x:100,y:515,
p:200, minS:1, maxS:4, vals:2, desimal:2, label:"rad/s"}
27. var slider2 = {tipe:"H", nama:"radius", x:420,y:515,
p:200, minS:0.5, maxS:1.2, vals:1.2, desimal:2,
label:"m"}
28.
29. var pegas1 = {x1:xBase, y1:yBase+150,
x2:xBase+0.95*(xPos-xBase), y2:yBase+150, putaran:10,
lebar:20, offset:10, warna1:"#2e2e2e", warna2:"#b7b7b7",
tebal:"5"}
30. var pegas2 = {x1:xSupport+150, y1:ySupport-180,
x2:xSupport+150, y2:ySupport-180+1.0*(yPos-(ySupport-
180)), putaran:10, lebar:20, offset:10, warna1:"#2e2e2e",
warna2:"#b7b7b7", tebal:"5"}
31.
32. var graf = {startX:xGraph, startY:yGraph, dataW:5,
dataH:8, tileW:40, skalaX:2, skalaY:0.4, desimalX:0,
desimalY:1, offsetX:0, offsetY:0, xLabel:'t (s)',
yLabel:'y (m)', fontLabel:'12pt Calibri',
warnaBG:'#daf6fb', warnaGaris:'#000', warnaLabel:'#000'}
33.
34. function setSimulasi() {
35. if (time >= 100.0) simAktif = 0;
36. if (simAktif == 1) {
37. hapusLayar();
38. //menampilkan teks
39. teks("Gerak Melingkar", 0.5*(canvas.width), 40, 'bold
18pt Calibri', 'blue', 'center');

```

```

40. teks("Simulasi gerakan bola pada lintasan melingkar",
 0.5*(canvas.width), 60, "12pt Calibri", "#000",
 "center");
41.
42. //tombol control
43. tombol("Play", 150,560, 80, 30, "bold 11pt Calibri",
 "white", "black", "gray", "r");
44. tombol("Pause", 240,560, 80, 30, "bold 11pt Calibri",
 "white", "black", "gray", "r");
45. tombol("<< Step", 330,560, 80, 30, "bold 11pt
 Calibri", "white", "black", "gray", "r");
46. tombol("Step >>", 420,560, 80, 30, "bold 11pt
 Calibri", "white", "black", "gray", "r");
47. tombol("Reset", 510,560, 80, 30, "bold 11pt Calibri",
 "white", "black", "gray", "r");
48.
49. //slider
50. teks("Omega = "+omega+" rad/s", slider1.x-15,
 slider1.y-20, "bold 13pt Calibri", "black", "left");
51. slider(slider1);
52. teks("Radius = "+length+" m", slider2.x-15,
 slider2.y-20, "bold 13pt Calibri", "black", "left");
53. slider(slider2);
54.
55. // pusat lingkaran dan lintasan
56. lingkaran(xSupport, ySupport, radius, 2, "black",
 "black");
57. lingkaran(xSupport, ySupport, length*rodLength, 1,
 "black", "none");
58. for (i=0; i<=5; i++) {
59. garis(xSupport-length*rodLength*
 Math.sin(30*i*Math.PI/180), ySupport-length*rodLength*
 Math.cos(30*i*Math.PI/180), xSupport+length*rodLength*
 Math.sin(30*i*Math.PI/180), ySupport+length*rodLength*
 Math.cos(30*i*Math.PI/180), 1, "black");
60. }
61.
62. // bola yang bergerak melingkar
63. index = index + 1;
64. time = index/100.0;
65. xPos = xSupport + length*rodLength*
 Math.cos(omega*time);
66. yPos = ySupport - length*rodLength*
 Math.sin(omega*time);
67. lingkaran(xPos,yPos, 2*radius, 1, "red", "red");
68.
69. // garis indikator Horisontal
70. for (i = -2; i<=2; i++) {
71. garis(xSupport+60*i, yBase+130, xSupport+60*i,
 yBase+170);
72. }
73.
74. // Bola pada pegas dengan Gerakan Harmonik Sederhana
75. lingkaran(xPos, yBase+150, 2*radius, 2, "black",
 "#f6f");
76.

```

```

77. // Tumpuan Pegas dan Pegas Horisontal
78. garis(xBase, yBase+130, xBase, yBase+170, 4);
79. pegas1.x2 = xBase+(xPos-xBase)-2*radius;
80. pegas(pegas1);
81.
82. // garis indikator vertikal
83. for (i = -2; i<=2; i++) {
84. garis(xSupport+130, ySupport+60*i, xSupport+170,
85. ySupport+60*i);
86. }
87. // Bola pada pegas dengan Gerakan Harmonik Sederhana
88. lingkaran(xSupport+150, yPos, 2*radius, 2, "black",
89. "#6ff");
90. // tumpuan Pegas dan Pegas vertikal
91. garis(xSupport+130, ySupport-180,xSupport+170,
92. ySupport-180, 4);
93. pegas2.y2=ySupport-180+1.0*(yPos-(ySupport-180)-
94. 2*radius);
95. pegas(pegas2);
96. // Grafik untuk menampilkan gerakan bola
97. grafik(graf);
98. konten.strokeStyle = "red";
99. konten.beginPath();
100. konten.moveTo(xGraph, yGraph+40*4);
101. var maxIndex = 1000;
102. if (maxIndex > index) maxIndex = index;
103. for (var ival = 1; ival <=maxIndex; ival++) {
104. konten.lineTo(xGraph+ival/5, ySupport -
105. length*rodLength*Math.sin(omega*ival/100));
106. }
107. konten.stroke();
108. var timeLabel = 'waktu (t) = ';
109. timeLabel = timeLabel + time.toFixed(2) + ' s';
110. teks(timeLabel, 120,100);
111. }
112. function mouseDown(event){
113. canvas.onmousemove = mouseDrag;
114. }
115.
116. function mouseDrag(event){
117. //prosedur mengecek slider
118. var sliderAktif = cekSlider(event);
119. if (sliderAktif != null){
120. if (sliderAktif.nama == "omega") {
121. omega = Number(sliderAktif.valS);
122. reset();
123. }

```

```

124. if (sliderAktif.nama == "radius") {
125. length = Number(sliderAktif.vals);
126. reset();
127. }
128. }
129. }
130.
131. function mouseUp(event){
132. //prosedure mengecek tombol
133. var tombolAktif = cekTombol(event);
134. if (tombolAktif != ""){
135. if (tombolAktif == "Play"){
136. window.clearTimeout(timer);
137. simAktif = 1;
138. jalankanSimulasi();
139. }
140. if (tombolAktif == "Reset")reset();
141. if (tombolAktif == "Pause"){
142. window.clearTimeout(timer);
143. simAktif = 0;
144. }
145. if (tombolAktif == "<< Step"){
146. window.clearTimeout(timer);
147. index -= 2;
148. if (index < -1) index = -1;
149. simAktif = 1;
150. setSimulasi();
151. }
152. if (tombolAktif == "Step >>"){
153. window.clearTimeout(timer);
154. simAktif = 1;
155. setSimulasi();
156. }
157. }
158. canvas.onmousemove = null;
159. }
160.
161. function reset() {
162. window.clearTimeout(timer);
163. index = -1;
164. time = 0.0;
165. omega = Number(slider1.vals);
166. length = Number(slider2.vals);
167. simAktif = 1;
168. setSimulasi();
169. }
170.
171. function jalankanSimulasi() {
172. setSimulasi();
173. if (simAktif == 1) {

```

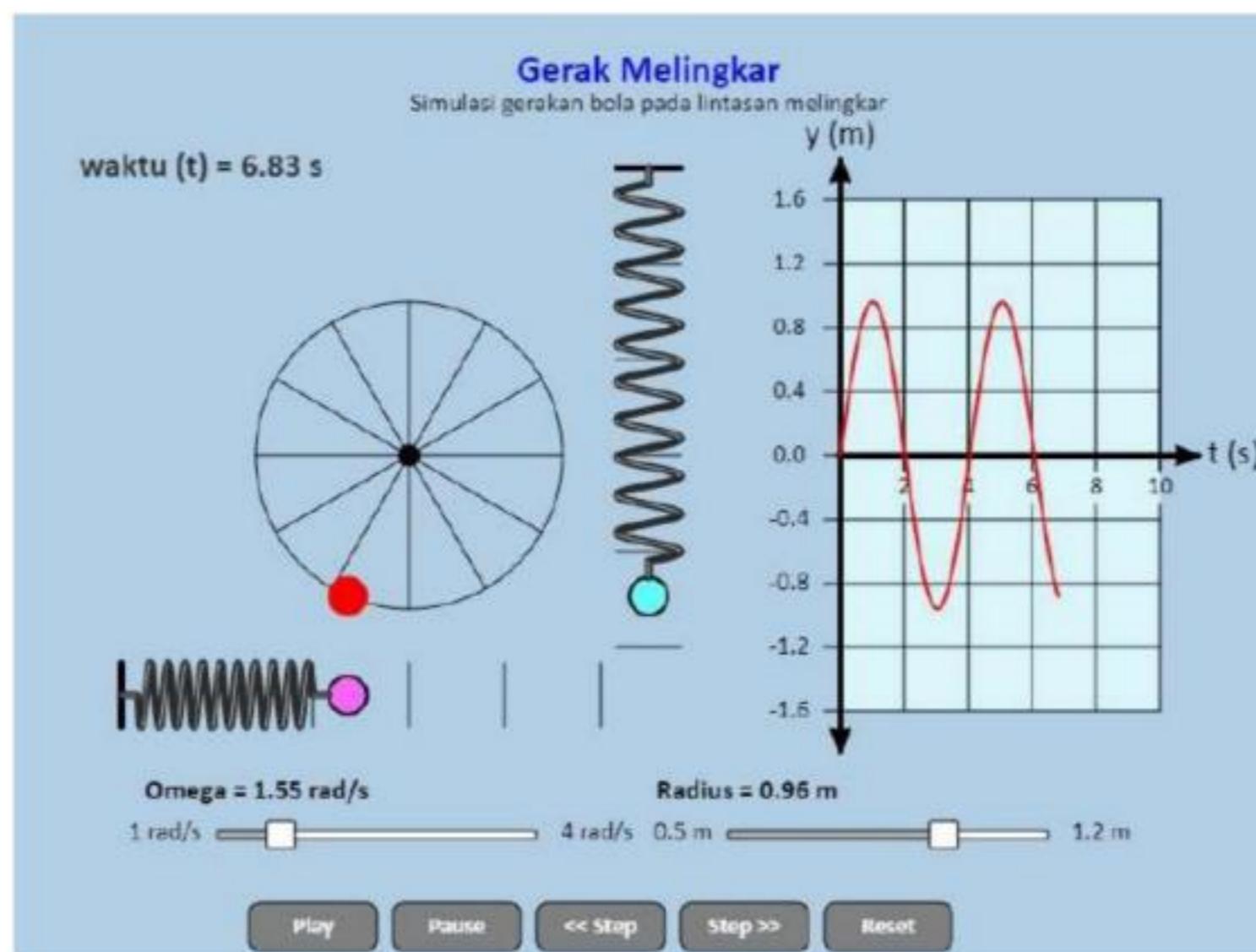
```

174. timer = window.setTimeout(jalankanSimulasi, 1);
175. }
176. }
177.
178. setSimulasi();

```

3. Simpan dengan nama **simulasi-12.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-12.

Jalankan *file* **tutorial-12.html**, dengan cara membukanya di internet *browser*. Maka kode di atas akan menghasilkan simulasi gerak melingkar sebagai berikut:



Gambar 9.6 Hasil simulasi gerak melingkar

#### d. Penjelasan Program

Tahapan untuk menghasilkan simulasi gerak melingkar adalah sebagai berikut :

1. Tentukan variabel yang dibutuhkan dalam perhitungan gerak melingkar, yaitu kecepatan sudut (*omega*), radius atau jarak objek ke pusat (*length*), penskalaan ukuran / transformasi linear ke sistem koordinat layar (*rodLength*).
2. Perhitungan kecepatan objek menggunakan trigonometri, dengan mengalikan kecepatan sudut (*omega*) dengan waktu (*time*).

```

xPos = xSupport + length*rodLength*Math.cos(omega*time);
yPos = ySupport - length*rodLength*Math.sin(omega*time);
lingkaran(xPos,yPos, 2*radius, 1, "red", "red");

```

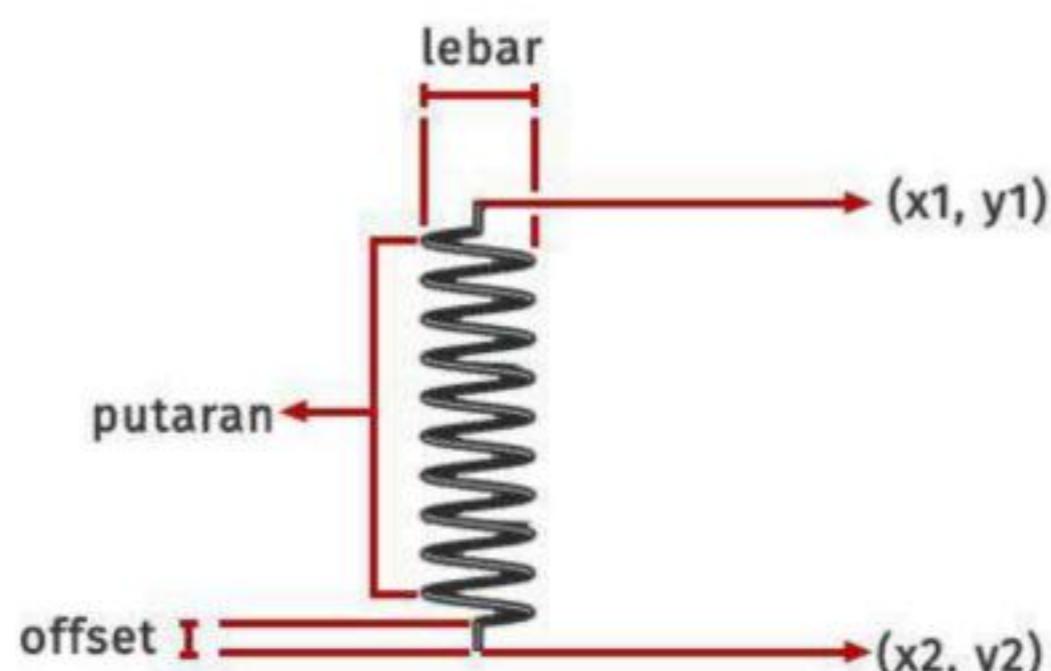
3. Untuk menunjukkan bahwa gerakan melingkar pada dasarnya merupakan gerak harmonik sederhana secara vertikal dan horizontal, maka ditampilkan pegas yang meregang secara dinamis mengikuti gerakan berputar.

Untuk mendefinisikan pegas digunakan variabel bertipe objek, yang selanjutnya posisi  $x2$  atau  $y2$  pegas diubah, sehingga pegas dapat meregang atau merapat.

```
pegas1.x2 = xBase+(xPos-xBase)-2*radius;
pegas(pegas1);
```

Adapun struktur data untuk membentuk pegas adalah sebagai berikut :

|         |                                                             |
|---------|-------------------------------------------------------------|
| x1      | Kordinat x ujung awal dari pegas                            |
| y1      | Kordinat y ujung awal dari pegas                            |
| x2      | Kordinat x ujung akhir dari pegas                           |
| y2      | Kordinat y ujung akhir dari pegas                           |
| putaran | Jumlah putaran pegas                                        |
| lebar   | Lebar pegas dalam satuan <i>pixel</i>                       |
| offset  | Jarak garis lurus di awal dan diakhir pegas sebelum putaran |
| warna1  | Warna terang pada pegas                                     |
| warna2  | Warna gelap pada pegas                                      |
| tebal   | Ketebalan garis pegas                                       |



Gambar 9.6 Struktur data fungsi pegas.



## BAB 10

### Simulasi Gaya

#### 10.1 Simulasi Hukum Newton

Hubungan antara gaya yang bekerja pada suatu benda dan gerak yang disebabkannya telah dijabarkan secara detail pada hukum Newton. Pada Bab ini akan dijelaskan tentang penerapan hukum Newton kedua, yaitu suatu benda akan bertambah kecepatannya jika diberikan gaya total yang arahnya sama dengan arah gerak benda. Sebaliknya, jika arah gaya total yang diberikan pada benda tersebut berlawanan dengan arah gerak benda maka gaya tersebut akan memperkecil kecepatan benda atau bahkan menghentikannya. Simulasi yang akan ditampilkan memiliki variabel masa benda dan energi yang diberikan, yang selanjutnya ditemukan variabel percepatan yang menggerakkan benda.

Secara sederhana rumus yang digunakan adalah

$$\Sigma F = m \cdot a$$

Dalam simulasi, kita dapat menambahkan perhitungan yang lebih kompleks dengan menambahkan variabel gesekan. Untuk lebih detailnya, perhatikan langkah berikut :

#### A. Tutorial 13 – Hukum Newton tentang Gerak

##### a. File HTML

1. Seperti pada projek tutorial sebelumnya, buatlah sebuah *folder* khusus dengan nama **tutorial-13**. *Copy file tutorial-12.html* dan *paste* ke dalam *folder*.
2. *Rename* nama *file* menjadi **tutorial-13.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
12. <script src="simulasi-13.js"></script>
```

3. Simpan kembali file HTML.

##### b. File library

1. *Copy paste* file **simulasiDasar.js** ke dalam *folder* **tutorial-13**.

### c. File simulasi-13.js

1. Pada Notepad++, buatlah sebuah file baru

2. Ketikan kode berikut :

```
1. aturCanvas();
2. setJudul("Hukum Newton 2");
3. hapusLayar("#b3cfe5");
4.
5. //listener untuk membaca event mouse
6. canvas.onmousedown = mouseDown;
7. canvas.onmouseup = mouseUp;
8.
9. var slider1 = {tipe:"H", nama:"massa", x:100,y:300, p:200,
 minS:1, maxS:5, valS:2.0, desimal:1, label:"Kg"}
10. var slider2 = {tipe:"H", nama:"gaya", x:100,y:360, p:200,
 minS:-20, maxS:20, valS:0.0, desimal:1, label:"N"}
11. var slider3 = {tipe:"H", nama:"friksi", x:100,y:420, p:200,
 minS:0, maxS:1, valS:0.0, desimal:3, label:" "}
12.
13. var index = -1;
14. var xBase = 300;
15. var yBase = 200;
16. var time = 0.0;
17. var x = 0.0;
18. var v = 0.0;
19. var a;
20. var m = 2.0;
21. var mu = 0;
22. var boxSize = 40*Math.sqrt(m);
23. var friction = 0;
24. var force = 0.0;
25. var xPos;
26. var yPos;
27. var timer;
28. var simAktif = 1;
29. var bergerak = 0;
30.
31. function setSimulasi(){
32. if ((time >= 10000.0)) simAktif = 0;
33. if (simAktif == 1) {
34. //hapus layar
35. hapusLayar();
36.
37. //menampilkan teks
38. teks("Gaya dan Gesekan", 0.5*(canvas.width), 50, 'bold
 16pt Calibri', 'blue', 'center');
39. teks("Massa = "+m+" Kg", 100, 280, 'bold 14pt Calibri',
 'black', 'left');
40. slider(slider1);
41. teks("Gaya = "+force+" N", 100, 340, 'bold 14pt
 Calibri', 'black', 'left');
42. slider(slider2);
```

```

43. teks("Friksi = "+mu, 100, 400, 'bold 14pt Calibri',
 'black', 'left');
44. slider(slider3);
45.
46. tombol("Reset", 450,300, 120, 40, "bold 12pt Calibri",
 "white", "black", "gray", "r");
47.
48. index = index + 1;
49. time = index/100.0;
50. oldv = v;
51. if (index > 0) {
52. a = (force+friction)/m;
53. v = v + a/100;
54. x = x + v/100;
55. }
56. //objek keluar pada sisi lainnya
57. if (x > 10) {
58. x = x - 20;
59. }else if (x < -10) {
60. x = x + 20;
61. }
62.
63. xPos = canvas.width/2 + 40*x;
64. boxSize = 40*Math.sqrt(m);
65. // kotak beban
66. kotak(xPos-boxSize/2, yBase-boxSize, boxSize, boxSize,
 1, "#000", "#6af");
67. // lantai
68. garis(0,yBase,canvas.width,yBase, 3, "#000");
69. // normal
70. panah(xPos,yBase-boxSize/2-10,0,4*m, 5, 3, "blue");
71. teks("F", xPos + 16, yBase-boxSize-20, "bold 20pt
 Calibri", "blue");
72. teks("N", xPos + 28, yBase-boxSize-12, "bold 14pt
 Calibri", "blue");
73. // gravitasi (massa)
74. panah(xPos,yBase-boxSize/2+10,0,-4*m, 5, 3,"#282");
75. teks("F", xPos + 16, yBase+20, "bold 20pt Calibri",
 "#282");
76. teks("g", xPos + 28, yBase+28, "bold 14pt Calibri",
 "#282");
77. // panah gaya
78. if (force > 0) {
79. var offset = 10;
80. } else {
81. var offset = -10;
82. }
83. panah(xPos+offset,yBase-boxSize/2,0.4*force,0, 5,
 3,"#c00");
84. if (Math.abs(force) > 0.01) {
85. teks("F", xPos + offset + (5+2*Math.abs(force))*_
 force/Math.abs(force), yBase-boxSize/2+12, "bold 20pt
 Calibri", "#c00");
86. }

```

```

87. if (bergerak == 0) {
88. friction = mu*10*m;
89. if (friction >= Math.abs(force)) {
90. friction = -force;
91. v = 0;
92. } else {
93. bergerak = 1;
94. friction = mu*10*m;
95. if (force > 0) friction = -friction;
96. }
97. }else {
98. friction = mu*10*m;
99. if ((v > 0) && (2 > 0)) friction = -friction;
100. if (((v > 0) && (oldv < 0)) || ((v < 0) &&
101. (oldv > 0))) {
102. if (Math.abs(friction) >= Math.abs(force)) {
103. bergerak = 0;
104. v = 0.0;
105. friction = -force;
106. }
107. }
108. //gesekan / friksi
109. if (friction > 0) {
110. offset = 10;
111. } else {
112. offset = -10;
113. }
114. panah(xPos + offset,yBase-boxSize/2,0.4*friction,0, 5,
115. 3,"black");
116. if (Math.abs(friction) > 0.01) {
117. teks("F", xPos + offset + (5+2*Math.abs(friction))*friction/Math.abs(friction), yBase-boxSize/2-20, "bold 20pt Calibri", "black");
118. teks("f", 8 + xPos + offset + (5+2*Math.abs(friction))*friction/Math.abs(friction), yBase-boxSize/2-12,"bold 14pt Calibri", "black");
119. }
120. var timeLabel = 'Waktu (t) = ';
121. timeLabel = timeLabel + time.toFixed(2) + ' s';
122. teks(timeLabel, 450, 280, "bold 14pt Calibri", "black",
123. "left");
124. }
125. function mouseDown(event){
126. canvas.onmousemove = mouseDrag;
127. }
128.
129. function mouseUp(event){
130. //prosedure mengecek tombol
131. var tombolAktif = cekTombol(event);
132. if (tombolAktif != ""){

```

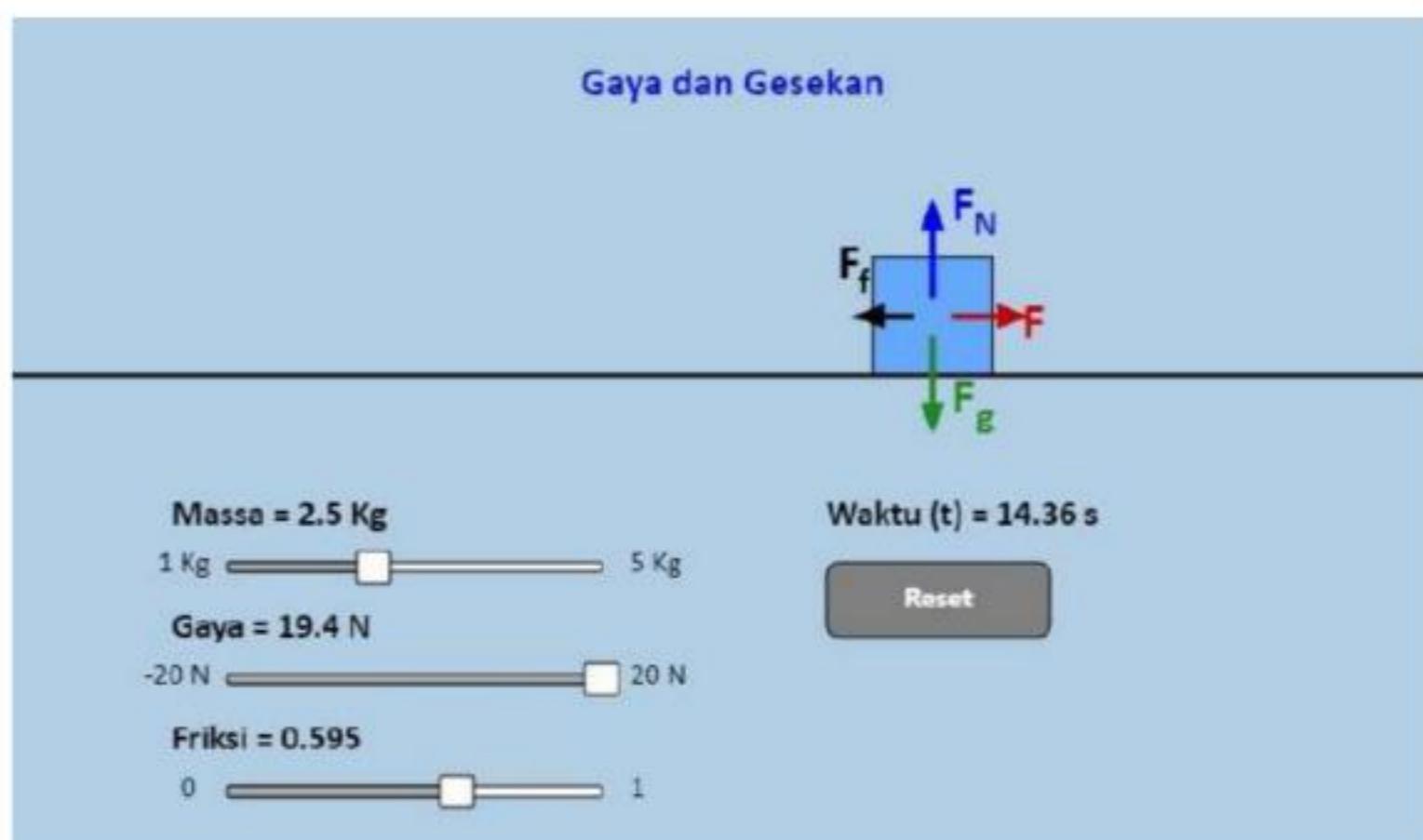
```

133. if (tombolAktif == "Reset") reset();
134. }
135. //menetralisir drag
136. canvas.onmousemove = null;
137. }
138.
139. function mouseDrag(event){
140. //prosedur mengecek slider
141. var sliderAktif = cekSlider(event);
142. if (sliderAktif != null){
143. if (sliderAktif.nama == "massa")
144. m = Number(sliderAktif.vals);
145. if (sliderAktif.nama == "gaya")
146. force = Number(sliderAktif.vals);
147. if (sliderAktif.nama == "friksi")
148. mu = Number(sliderAktif.vals);
149. }
150. }
151. function jalankanSimulasi() {
152. setSimulasi();
153. if (simAktif == 1) {
154. timer = window.setTimeout(jalankanSimulasi, 10);
155. }
156. function reset(){
157. window.clearTimeout(timer);
158. index = -1;
159. time = 0.0;
160. x = 0;
161. v = 0;
162. slider1.vals = 2;
163. m = 2;
164. slider2.vals = 0;
165. force = 0;
166. slider3.vals = 0;
167. mu = 0;
168. simAktif = 1;
169. jalankanSimulasi();
170. }
171.
172. jalankanSimulasi();

```

3. Simpan dengan nama **simulasi-13.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-13.

Jalankan file **tutorial-13.html**, dengan cara membukanya di *internet browser*. Maka kode di atas akan menghasilkan simulasi gerak dengan vektor gaya sesuai dengan hukum Newton sebagai berikut :



Gambar 10.1 Hasil tutorial Simulasi Hukum Newton

#### d. Penjelasan Program

Seperti pada penjelasan tutorial sebelumnya, untuk membuat simulasi di atas, pada dasarnya tahapan yang dilakukan adalah menggerakkan objek visual sesuai dengan rumus yang diterapkan. Dalam hal ini digunakan rumus :

$$\Sigma F = m \cdot a$$

Dimana nilai Force/gaya ( $F$ ) dan nilai massa ( $m$ ) bersifat dinamis dan dapat diubah melalui *slider*. Sehingga, kita dapat mencari nilai percepatan ( $a$ ). Adapun rumus tersebut diletakkan pada baris 51 – 55.

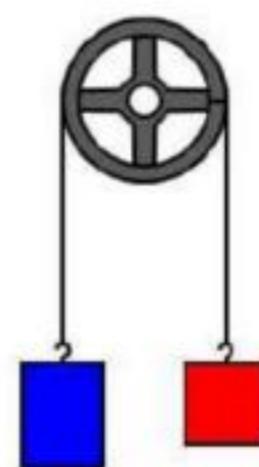
```
55. if (index > 0) {
56. a = (force+friction)/m;
57. v = v + a/100;
58. x = x + v/100;
59. }
```

Setelah mengetahui nilai  $a$  (baris 56), maka kita dapat menentukan kecepatan objek ( $v$ ) dan pergeserannya pada kordinat  $x$  (baris 58). Perlu diketahui bahwa pada kode baris 56, variabel *friction* bernilai negatif, sehingga kode tersebut pada dasarnya bernilai  $a = (force-friction)/m$ ;

Pada baris selanjutnya, kordinat  $x$  dipindah ketika keluar dari layar, sehingga menghasilkan gerakan objek yang berulang (*looping*) dari satu sisi ke sisi lainnya.

## B. Tutorial 14 – Mesin Atwood (Dinamika Translasi)

Pada tutorial selanjutnya, disimulasikan tentang keseimbangan antara 2 beban yang terkait pada sebuah sistem katrol sederhana (mesin Atwood). Pada simulasi ini pengguna dapat mengetahui hubungan antara massa benda dan percepatannya. Secara sederhana simulasi yang dimaksud digambarkan sebagai berikut :



Gambar 10.2 Simulasi keseimbangan beban pada mesin Atwood.

Dalam simulasi di atas dua benda bermassa  $m_1$  dan  $m_2$  digantungkan pada katrol tetap. Jika  $m_2 > m_1$  dan benda 1 bergerak ke atas serta benda 2 bergerak ke bawah dengan percepatan  $a$ , maka rumus percepatan pada sistem ini adalah sebagai berikut:

$$a = g (m_2 - m_1) / (m_2 + m_1)$$

dengan mengetahui nilai percepatan, maka kita dapat menentukan arah gerak dan besarnya jarak yang dilakukan oleh masing-masing benda dan sudut putaran katrol. Untuk lebih jelasnya, perhatikan tutorial berikut :

### a. *File HTML*

1. Seperti pada projek tutorial sebelumnya, buatlah sebuah *folder* khusus dengan nama **tutorial-14**. *Copy file tutorial-13.html* dan *paste* ke dalam *folder*.
2. *Rename* nama *file* menjadi **tutorial-14.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
12. <script src="simulasi-14.js"></script>
```

3. Simpan kembali *file* HTML.

### b. *File library*

1. *Copy paste* *file* **simulasiDasar.js** ke dalam *folder* **tutorial-14**.

### c. File simulasi-14.js

1. Pada Notepad++, buatlah sebuah file baru

2. Ketikkan kode berikut :

```
1. aturCanvas();
2. setJudul("Keseimbangan Beban");
3. hapusLayar("#b3cfe5");
4.
5. //listener untuk membaca event mouse
6. canvas.onmousedown = mouseDown;
7. canvas.onmouseup = mouseUp;
8.
9. var index = -1;
10. var xBase = canvas.width/2;
11. var yBase = 120;
12. var xPos = 0.0;
13. var y1PosInit = 3; //panjang tali
14. var y2PosInit = 3;
15. var scaleFactor = 40;
16. var mass1 = 1.5;
17. var mass2 = 1.0;
18. var g = 10;
19. var a = g*(mass1-mass2) / (mass1+mass2);
20. var radius = 40;
21. var time = -0.5;
22. var timer;
23. var simAktif = 1;
24. var theta = 0.0;
25.
26. var slider1 = {tipe:"H", nama:"massa1", x:100,y:515,
27. p:200, minS:0.1, maxS:2.0, valS:1.5, desimal:2,
28. label:"kg"}
29. var slider2 = {tipe:"H", nama:"massa2", x:420,y:515,
30. p:200, minS:0.1, maxS:2.0, valS:1.0, desimal:2,
31. label:"kg"}
32.
33. var katrol = {cx:xBase, cy:yBase, radius0:radius,
34. radiusH:5, tbl:8}
35.
36. function setSimulasi() {
37. if ((index >= 1000) || (Math.abs(0.5*a*time*time) >=
38. 2.4)) simAktif = 0;
39. if (simAktif == 1) {
40. hapusLayar();
```

```
41. //menampilkan teks
42. teks("Keseimbangan Beban", 0.5*(canvas.width), 40,
43. 'bold 18pt Calibri', 'blue', 'center');
44. teks("Simulasi keseimbangan beban pada katrol",
45. 0.5*(canvas.width), 60, "12pt Calibri", "#000",
46. "center");
47.
48. //tombol control
49. tombol("Play", 150,560, 80, 30, "bold 11pt")
```

```

 Calibri", "white", "black", "gray", "r");
41. tombol("Pause", 240,560, 80, 30, "bold 11pt
 Calibri", "white", "black", "gray", "r");
42. tombol("<< Step", 330,560, 80, 30, "bold 11pt
 Calibri", "white", "black", "gray", "r");
43. tombol("Step >>", 420,560, 80, 30, "bold 11pt
 Calibri", "white", "black", "gray", "r");
44. tombol("Reset", 510,560, 80, 30, "bold 11pt
 Calibri", "white", "black", "gray", "r");
45.
46. //slider
47. teks("Massa 1 = "+mass1+" kg", slider1.x-15,
 slider1.y-20, "bold 13pt Calibri", "black",
 "left");
48. slider(slider1);
49. teks("Massa 2 = "+mass2+" kg", slider2.x-15,
 slider2.y-20, "bold 13pt Calibri", "black",
 "left");
50. slider(slider2);
51.
52. //simulasi
53. index = index + 1;
54. time = index/100.0;
55. var y1Pos = y1PosInit + 0.5*a*time*time;
56. var y2Pos = y2PosInit - 0.5*a*time*time;
57. theta = 0.5*a*time*time*scaleFactor/radius;
58. var sudut = -theta*180/Math.PI;
59.
60. // katrol
61. roda(katrol, sudut);
62.
63. // tali
64. garis(xBase-radius, yBase, xBase-radius, yBase +
 scaleFactor*y1Pos, 2, "black");
65. garis(xBase+radius, yBase, xBase+radius, yBase +
 scaleFactor*y2Pos, 2, "black");
66. //pengait
67. pengait(xBase-radius, yBase+scaleFactor*y1Pos, 9,
 2);
68. pengait(xBase+radius, yBase+scaleFactor*y2Pos, 9,
 2);
69. // beban 1
70. kotak(xBase-radius-20, yBase + scaleFactor*y1Pos+9,
 40, 40*mass1,2, "black", "blue");
71. // beban 2
72. kotak(xBase+radius-20, yBase + scaleFactor*y2Pos+9,
 40, 40*mass2, 2, "black", "red");
73.
74. var fT1 = mass1*(g-a);
75. var fT2 = mass2*(g+a);
76.
77. var aLabel = 'a = ';
78. aLabel = aLabel + (a).toFixed(2) + ' m/s2';
79. teks(aLabel, 100, 120, "bold 13pt Calibri",
 "black", "left");

```

```

80. var mg1Label = 'Mg = ';
81. mg1Label = mg1Label + (mass1*g).toFixed(2) + ' N';
82. teks(mg1Label, 100, 160, "bold 13pt Calibri",
83. "black", "left");
84. var fT1Label = 'F = ';
85. fT1Label = fT1Label + (fT1).toFixed(2) + ' N';
86. teks(fT1Label, 100, 200, "bold 13pt Calibri",
87. "black", "left");
88. teks("T1", 100+8, 208, "bold 11pt Calibri",
89. "black", "left");
90. var mg2Label = 'mg = ';
91. mg2Label = mg2Label + (mass2*g).toFixed(2) + ' N';
92. teks(mg2Label, 100, 240, "bold 13pt Calibri",
93. "black", "left");
94. var fT2Label = 'F = ';
95. fT2Label = fT2Label + (fT2).toFixed(2) + ' N';
96. teks(fT2Label, 100, 280, "bold 13pt Calibri",
97. "black", "left");
98. }
99.
100. function mouseDown(event){
101. canvas.onmousemove = mouseDrag;
102. }
103.
104. function mouseDrag(event){
105. //prosedur mengecek slider
106. var sliderAktif = cekSlider(event);
107. if (sliderAktif != null){
108. if (sliderAktif.nama == "massal") {
109. mass1 = Number(sliderAktif.vals);
110. reset();
111. }
112. if (sliderAktif.nama == "massa2") {
113. mass2 = Number(sliderAktif.vals);
114. reset();
115. }
116. }
117. }
118.
119. function mouseUp(event){
120. //prosedure mengecek tombol
121. var tombolAktif = cekTombol(event);
122. if (tombolAktif != ""){
123. if (tombolAktif == "Play"){
124. window.clearTimeout(timer);
125. simAktif = 1;
126. jalankanSimulasi();

```

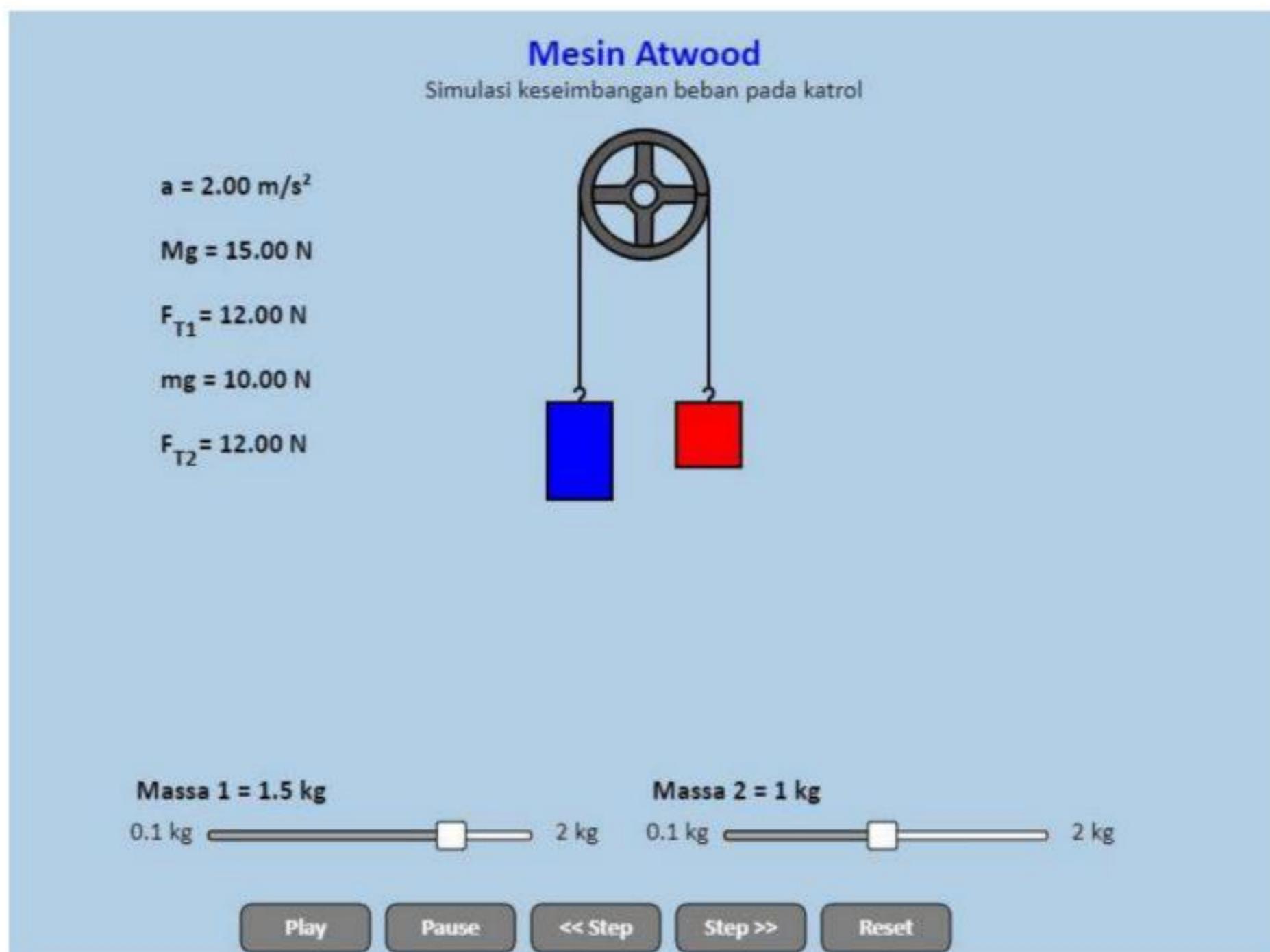
```

127. }
128. if (tombolAktif == "Reset") reset();
129. if (tombolAktif == "Pause"){
130. window.clearTimeout(timer);
131. simAktif = 0;
132. }
133. if (tombolAktif == "<< Step"){
134. window.clearTimeout(timer);
135. index -= 2;
136. if (index < -1) index = -1;
137. simAktif = 1;
138. setSimulasi();
139. }
140. if (tombolAktif == "Step >>"){
141. window.clearTimeout(timer);
142. simAktif = 1;
143. setSimulasi();
144. }
145. }
146. canvas.onmousemove = null;
147. }
148.
149. function reset() {
150. window.clearTimeout(timer);
151. index = -1;
152. time = 0;
153. a = g*(mass1-mass2)/(mass1+mass2);
154. simAktif = 1;
155. setSimulasi();
156. }
157.
158. function jalankanSimulasi() {
159. setSimulasi();
160. if (simAktif == 1) {
161. timer = window.setTimeout(jalankanSimulasi, 1);
162. }
163. }
164.
165. setSimulasi();

```

3. Simpan dengan nama **simulasi-14.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-14.

Jalankan *file* **tutorial-14.html**, dengan cara membukanya di internet *browser*. Maka kode di atas akan menghasilkan simulasi keseimbangan antara 2 beban sebagai berikut :



Gambar 10.3 Hasil tutorial simulasi mesin Atwood

#### d. Penjelasan Program

Pada tutorial keseimbangan beban di atas, ketika terdapat perbedaan massa benda, maka akan terjadi percepatan yang menyebabkan beban bergerak secara GLBB (Gerak Lurus Berubah Beraturan), sehingga untuk menghitung pergeseran posisi 2 beban tersebut dalam hal ini digunakan rumus :

$$x = x_0 + v_0 \cdot t + (a \cdot t \cdot t)/2$$

Dimana posisi pergerakan pada beban tersebut dilakukan secara vertikal, sehingga nilai jarak ( $x$ ) di atas direpresentasikan dengan variabel  $y1Pos$ . Nilai percepatan ( $a$ ) diperoleh dengan menghitung perbandingan masa antara 2 beban (lihat baris 19 dan 155). Selanjutnya perhitungan simulasi dilakukan pada baris 53 – 58 menggunakan rumus GLBB di atas .

```

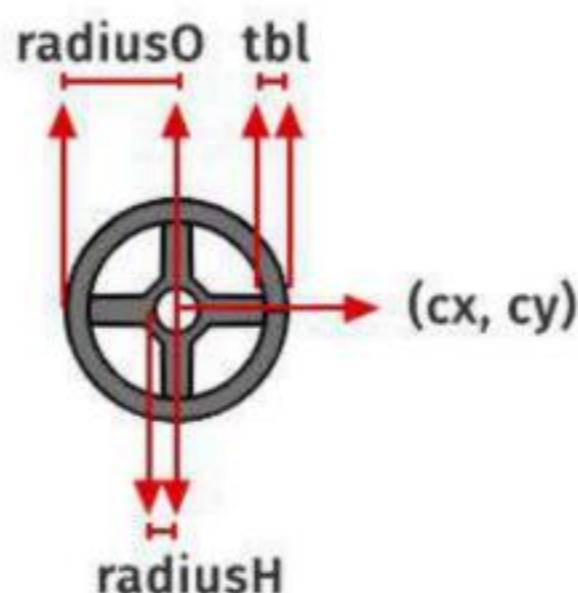
53. index = index + 1;
54. time = index/100.0;
55. var y1Pos = y1PosInit + 0.5*a*time*time;
56. var y2Pos = y2PosInit - 0.5*a*time*time;
57. theta = 0.5*a*time*time*scaleFactor/radius;
58. var sudut = -theta*180/Math.PI;
```

Perhatikan bahwa pada baris 57-58 dilakukan perhitungan sudut `theta`, yang digunakan untuk menggerakkan `roda` katrol pada baris 61.

```
61. roda(katrol, sudut);
```

Fungsi `roda` di atas, digunakan untuk membuat elemen roda dengan data variabel `katrol` (baris 29). Adapun struktur data untuk membuat roda adalah sebagai berikut :

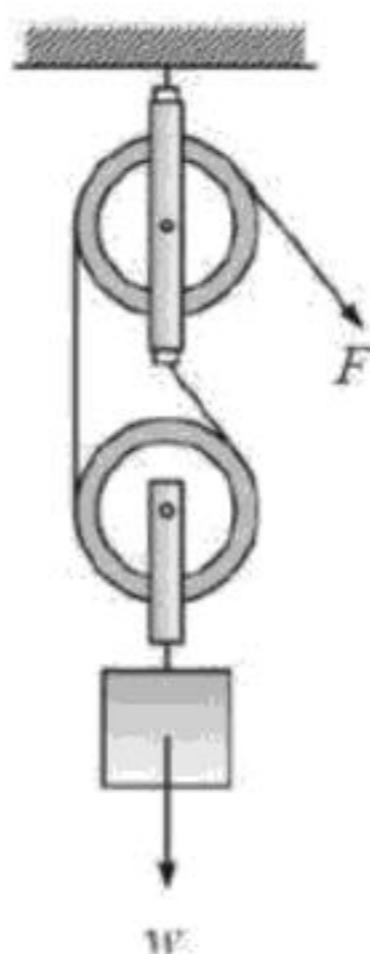
|                      |                                                               |
|----------------------|---------------------------------------------------------------|
| <code>cx</code>      | Kordinat x pusat roda                                         |
| <code>cy</code>      | Kordinat y pusat roda                                         |
| <code>radiusO</code> | Jari-jari luar dari roda                                      |
| <code>radiusH</code> | Jari-jari pusat roda                                          |
| <code>tbl</code>     | Ketebalan roda (jarak dari jari-jari luar ke jari-jari dalam) |



Gambar 10.4 Struktur data untuk membentuk roda

### C. Tutorial 15 – Sistem Katrol Ganda

Setelah memahami tutorial keseimbangan beban pada mesin Atwood di atas, kita dapat mengembangkannya lebih jauh menjadi sebuah simulasi sistem katrol sederhana, dalam hal ini kita dapat membuat sistem katrol ganda (katrol majemuk). Katrol ganda atau katrol kombinasi/majemuk merupakan gabungan antara katrol tetap dan katrol bergerak. Katrol kombinasi sering disebut *takal*. Dalam sebuah sistem katrol ganda terdiri atas  $n$  buah katrol, maka keuntungan mekanisnya dapat dicari dengan cara menghitung banyaknya gaya yang bekerja. Pada simulasi ini pengguna simulasi dapat mengatur massa beban, dan dapat menarik beban dengan sistem *drag mouse*. Secara sederhana simulasi memiliki tampilan visual sebagai berikut:



Gambar 10.5 Simulasi katrol ganda.

Untuk membuat simulasi katrol ganda, perhatikan langkah-langkah berikut :

**a. File HTML**

1. Seperti pada projek tutorial sebelumnya, buatlah sebuah *folder* khusus dengan nama **tutorial-15**. *Copy file tutorial-14.html* dan *paste* ke dalam *folder*.
2. *Rename* nama *file* menjadi **tutorial-15.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
12. <script src="simulasi-15.js"></script>
```

3. Simpan kembali *file* HTML.

**b. File library**

1. *Copy paste* *file* **simulasiDasar.js** ke dalam *folder* **tutorial-15**.

**c. File simulasi-15.js**

1. Pada Notepad++, buatlah sebuah *file* baru
2. Ketikkan kode berikut :

```
1. aturCanvas();
2. setJudul("Katrol Ganda");
3. hapusLayar("#b3cfe5");
4.
5. //listener untuk membaca event mouse
6. canvas.onmousedown = mouseDown;
7. canvas.onmouseup = mouseUp;
```

```

8. var xBase = canvas.width/2;
9. var yBase = 120;
10. var jarakKatrol = 200;
11. var beban = 1.5;
12. var radius = 40;
13. var timer;
14. var simAktif = 1;
15.
16. var slider1 = {tipe:"H", nama:"massa1", x:260,y:550,
17. p:200, minS:0.1, maxS:2.0, valS:1.5, desimal:2,
18. label:"kg"}
19. var katrolA = {cx:xBase, cy:yBase, radiusO:radius,
20. radiusH:5, tbl:8}
21. var katrolB = {cx:xBase, cy:yBase+jarakKatrol,
22. radiusO:radius, radiusH:5, tbl:8}
23.
24. var penarik = {nama:"penarik", x:550, y:250, w:40,
25. h:40, limit:"radius=90"};
26. var xStart = penarik.x;
27. var yStart = penarik.y;
28. var xP = penarik.x;
29. var yP = penarik.y;
30. var selisih;
31. var sudut = 0;
32.
33. function setSimulasi() {
34. if (simAktif == 1) {
35. hapusLayar();
36. //menampilkan teks
37. teks("Katrol Ganda", 0.5*(canvas.width), 40, 'bold
38. 18pt Calibri', 'blue', 'center');
39. teks("Simulasi katrol ganda", 0.5*(canvas.width),
40. 60, "12pt Calibri", "#000", "center");
41.
42. //slider
43. teks("Massa = "+bebant+ " kg", slider1.x-15,
44. slider1.y-20, "bold 13pt Calibri", "black",
45. "left");
46. slider(slider1);
47.
48. var sudutGaya = (cariSudut(katrolA.cx, katrolA.cy,
49. penarik.x, penarik.y)-90)*Math.PI/180;
50. // tali
51. garis(katrolA.cx-radius, katrolA.cy, katrolB.cx-
52. radius, katrolB.cy, 2, "black");
53. var sudutAB = -170;
54. garis(katrolB.cx+radius*Math.cos(sudutAB),
55. katrolB.cy+radius*Math.sin(sudutAB), katrolA.cx,
56. katrolA.cy+radius+19, 2, "black");
57. var xa = katrolA.cx+radius*Math.cos(sudutGaya);
58. var ya = katrolA.cy+radius*Math.sin(sudutGaya);
59. garis(xa, ya, penarik.x, penarik.y, 2, "black");
60.
61. }
62. }

```

```

49. // katrol
50. roda(katrolA, sudut);
51. roda(katrolB, sudut);
52.
53. //kotak pengait katrol
54. kotak(katrolA.cx - 5, katrolA.cy - radius - 10, 10,
55. 2*(radius+10));
56. pengait(katrolA.cx, katrolA.cy + radius+10, 9, 2,
57. 2);
58. kotak(katrolB.cx - 5, katrolB.cy - 10, 10,
59. radius+20);
60. pengait(katrolB.cx, katrolB.cy + radius+10, 9, 2);
61.
62. // beban 1
63. kotak(katrolB.cx-20, katrolB.cy + radius + 19, 40,
64. 40*bebani,2, "black", "blue");
65. // kotak gaya tarik
66. kotak(penarik.x, penarik.y, penarik.w, penarik.h,
67. 2, "black", "red");
68.
69. var jarakAwal = jarak(xStart, yStart, xa, ya);
70. var jarakTarikan = jarak(penarik.x, penarik.y, xa,
71. ya);
72. selisih = jarakTarikan-jarakAwal;
73. sudut = selisih;
74.
75. jarakKatrol = 200-selisih;
76. katrolB.cy = yBase+jarakKatrol;
77.
78. if (Math.abs(selisih)>90){
79. penarik.x = ox;
80. penarik.y = oy;
81. }
82.
83. function mouseDown(event){
84. canvas.onmousemove = mouseDrag;
85. startDrag(mouseDrag);
86. }
87.
88. function mouseDrag(event){
89. //prosedur mengecek penarik katrol
90. var drag = cekDrag(event);
91. if (drag != null)setSimulasi();
92. //prosedur mengecek slider
93. var sliderAktif = cekSlider(event);
94. if (sliderAktif != null){

```

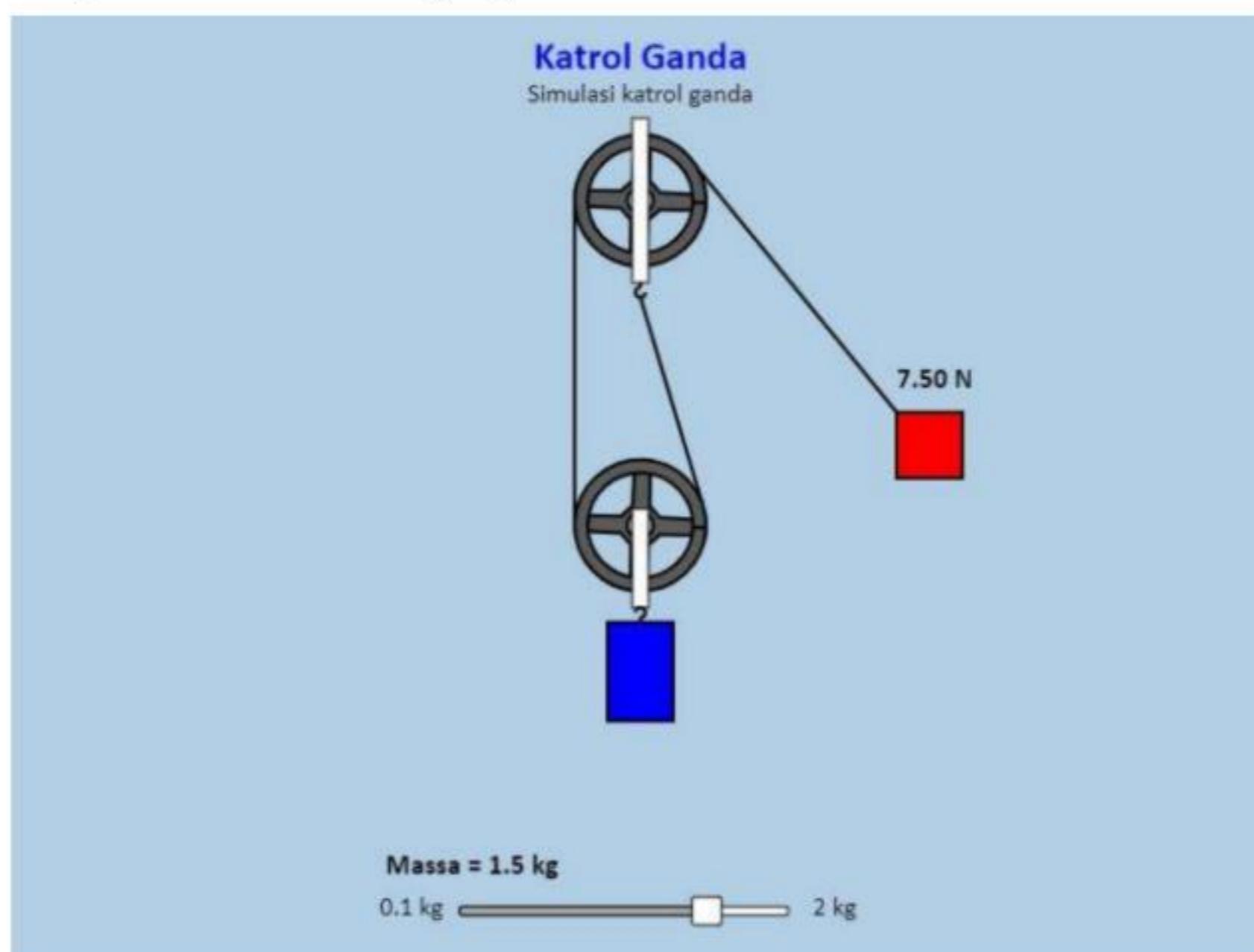
```

95. if (sliderAktif.nama == "massa1") {
96. beban = Number(sliderAktif.vals);
97. setSimulasi();
98. }
99. }
100. }
101.
102. function mouseUp(event) {
103. canvas.onmousemove = null;
104. }
105.
106. setDrag(penarik);
107. setSimulasi();

```

3. Simpan dengan nama **simulasi-15.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-15.

Jalankan *file* **tutorial-15.html**, dengan cara membukanya di *internet browser*. Maka kode di atas akan menghasilkan simulasi katrol ganda. Pada simulasi ini, pengguna dapat mengatur massa beban, dan menarik tali kuasa dengan cara *drag* kotak berwarna merah. Pada simulasi ini dihitung gaya kuasa adalah setengah dari massa beban yang ditentukan.



Gambar 10.6 Hasil tutorial simulasi katrol ganda

#### d. Penjelasan Program

Pada tutorial di atas, secara teknis urutan untuk membentuk simulasi adalah sebagai berikut :

1. Penentuan variabel yang digunakan untuk simulasi, dimana pada simulasi ini variabel utama adalah `jarakKatrol` dan `beban`.
2. Tentukan data variabel yang akan digunakan sebagai roda katrol, dalam hal ini terdapat 2 katrol yaitu `katrolA` dan `katrolB`.
3. Karena pada simulasi ini, pengguna dapat melakukan tarikan dengan teknik *drag*, maka perlu didefinisikan objek yang akan di *drag*, yaitu `penarik`. Perhatikan bahwa pada variabel `penarik`, ditentukan *limit* bernilai `radius=90`, sehingga pengguna hanya bisa menarik dan mengulur tali sejauh 90 *pixel* dari titik awal. Hal ini ditujukan untuk membatasi area agar simulasi logis untuk dijalankan.
4. Perhitungan utama dalam simulasi ini adalah dengan menghitung selisih jarak antara tali sebelum ditarik/diulur dan setelahnya. Selisih ini akan menentukan gerakan dari `katrolB`, sehingga dapat dilakukan perhitungan ulang untuk posisi masing-masing objek. Untuk menghitungnya digunakan fungsi `jarak` (menghitung jarak antara 2 titik kordinat)

```
64. var jarakAwal = jarak(xStart, yStart, xa, ya);
65. var jarakTarikan = jarak(penarik.x, penarik.y, xa, ya);
66. selisih = jarakTarikan-jarakAwal;
67. sudut = selisih;
```

5. Untuk menampilkan gaya kuasa, digunakan rumus sederhana

```
79. var fT = (beban*10)/2;
```

6. Pada tahapan selanjutnya, anda dapat menambahkan jumlah katrol atau menambahkan kompleksitas lainnya pada simulasi ini.

## BAB 11

### Simulasi Suhu

#### 11.1 Simulasi Konversi Suhu

Suhu didefinisikan sebagai ukuran atau derajat panas dinginnya suatu benda atau sistem. Suhu dapat didefinisikan juga sebagai ukuran energi kinetik rata-rata yang dimiliki oleh molekul-molekul suatu benda. Dalam laboratorium, untuk mengukur suhu secara umum digunakan termometer *bulb*. Termometer jenis ini memiliki gelembung besar (*bulb*) pada ujung bawah tempat menampung cairan, dan tabung sempit (lubang kapiler) untuk menekankan perubahan volume atau tempat pemuaian cairan. Dalam penggunaan termometer *bulb* praktikan harus melindungi *bulb* dari benturan dan menghindari pengukuran yang melebihi skala termometer, atas alasan inilah simulasi menggunakan termometer virtual akan mempermudah praktikan dalam memahami penggunaan termometer *bulb*.

Termometer *bulb* memiliki skala disepanjang tuba gelas yang menjadi tanda besaran temperatur yang secara umum dikenal dengan skala Celcius, Farenheit, Kelvin dan Reamur. Masing-masing skala memiliki perbandingan yang spesifik, sehingga sering kali diperlukan konversi dari skala satu ke skala yang lain. Pada tutorial selanjutnya, akan dibuat simulasi konversi suhu pada termometer *bulb*. Untuk lebih detailnya, perhatikan langkah berikut :

#### A. Tutorial 16 – Konversi Suhu

##### a. File HTML

1. Seperti pada projek tutorial sebelumnya, buatlah sebuah *folder* khusus dengan nama **tutorial-16**. *Copy file tutorial-15.html* dan *paste* ke dalam *folder*.
2. *Rename* nama *file* menjadi **tutorial-16.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
12. <script src="simulasi-16.js"></script>
```

3. Simpan kembali *file* HTML.

**b. File library**

1. Copy paste file **simulasiDasar.js** ke dalam folder **tutorial-16**.

**c. File simulasi-16.js**

1. Pada Notepad++, buatlah sebuah *file* baru.

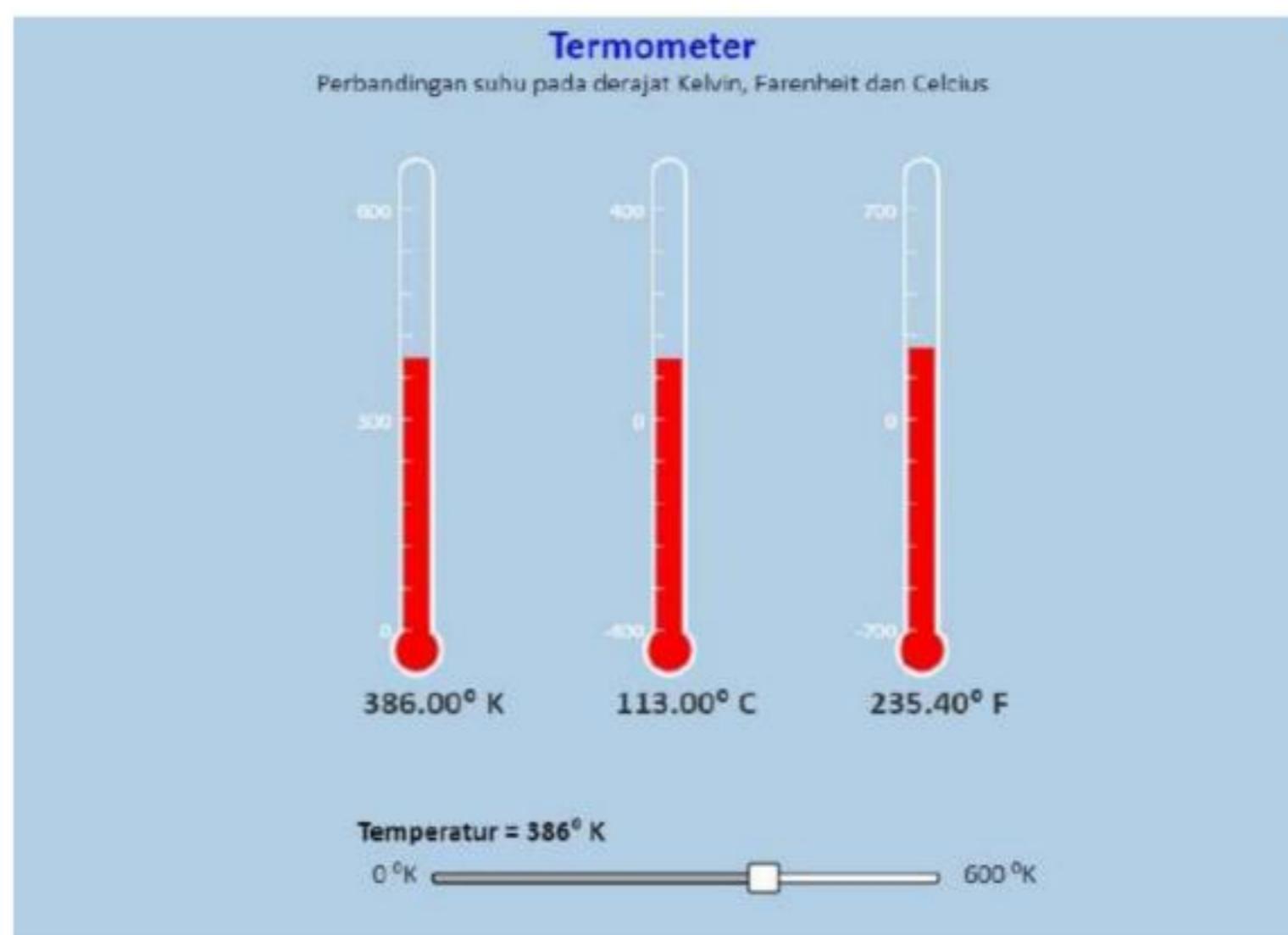
2. Ketikkan kode berikut :

```
1. aturCanvas();
2. setJudul("Termometer dan Konversi Suhu");
3. hapusLayar("#b3cfe5");
4.
5. //listener untuk membaca event mouse
6. canvas.onmousedown = mouseDown;
7. canvas.onmouseup = mouseUp;
8.
9. var suhuK = 300;
10. var suhuC = 27;
11. var suhuF = 80.6;
12.
13. var slider1 = {tipe:"H", nama:"suhu", x:240,y:525, p:300,
minS:0, maxS:600, valS:300, desimal:0, label:"0K"}
14.
15. var termo1 = {x:250, y:100, l:20, t:250, min:0, max:600,
val:300, offset: 15, warnaGaris: "#f8f8f8", warnaIsi:
"red",label:["0", "300", "600"]}
16. var termo2 = {x:400, y:100, l:20, t:250, min:-400,
max:400, val:27, offset: 15, warnaGaris: "#f8f8f8",
warnaIsi: "red",label:["-400", "0", "400"]}
17. var termo3 = {x:550, y:100, l:20, t:250, min:-700,
max:700, val:80.6, offset: 15, warnaGaris: "#f8f8f8",
warnaIsi: "red",label:["-700", "0", "700"]}
18.
19. function setSimulasi() {
20. hapusLayar();
21. //menampilkan teks
22. teks("Termometer", 0.5*(canvas.width), 40, 'bold 18pt
Calibri', 'blue', 'center');
23. teks("Perbandingan suhu pada derajat Kelvin, Farenheit
dan Celcius", 0.5*(canvas.width), 60, "12pt Calibri",
"#000", "center");
24.
25. //slider
26. teks("Temperatur = "+suhuK+"0 K", slider1.x-15,
slider1.y-20, "bold 13pt Calibri", "black", "left");
27. slider(slider1);
28.
29. //termometer
30. termo1.val = suhuK;
31. suhuC = suhuK-273;
32. termo2.val = suhuC;
33. suhuF = (suhuK-273)*1.8+32;
```

```
34. termo3.val = suhuF;
35. thermometer(termo1);
36. thermometer(termo2);
37. thermometer(termo3);
38.
39. //teks
40. teks(suhuK.toFixed(2)+"° K", termo1.x+20, 430);
41. teks(suhuC.toFixed(2)+"° C", termo2.x+20, 430);
42. teks(suhuF.toFixed(2)+"° F", termo3.x+20, 430);
43. }
44.
45. function mouseDown(event){
46. canvas.onmousemove = mouseDrag;
47. }
48.
49. function mouseDrag(event){
50. //prosedur mengecek slider
51. var sliderAktif = cekSlider(event);
52. if (sliderAktif != null){
53. if (sliderAktif.nama == "suhu") {
54. suhuK = Number(sliderAktif.vals);
55. setSimulasi();
56. }
57. }
58. }
59.
60. function mouseUp(event){
61. canvas.onmousemove = null;
62. }
63.
64. setSimulasi();
```

3. Simpan dengan nama **simulasi-16.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-16.

Jalankan *file* **tutorial-16.html**, dengan cara membukanya di *internet browser*. Maka kode di atas akan menghasilkan simulasi konversi suhu pada termometer *bulb* sebagai berikut :



Gambar 11.1 Hasil tutorial konversi suhu

#### d. Penjelasan Program

Simulasi pada tutorial 16 tersebut relatif sederhana karena tidak membutuhkan penerapan rumus yang kompleks. Untuk merubah suhu dari Kelvin ke skala lainnya digunakan rumus sederhana seperti pada baris 30-34.

```

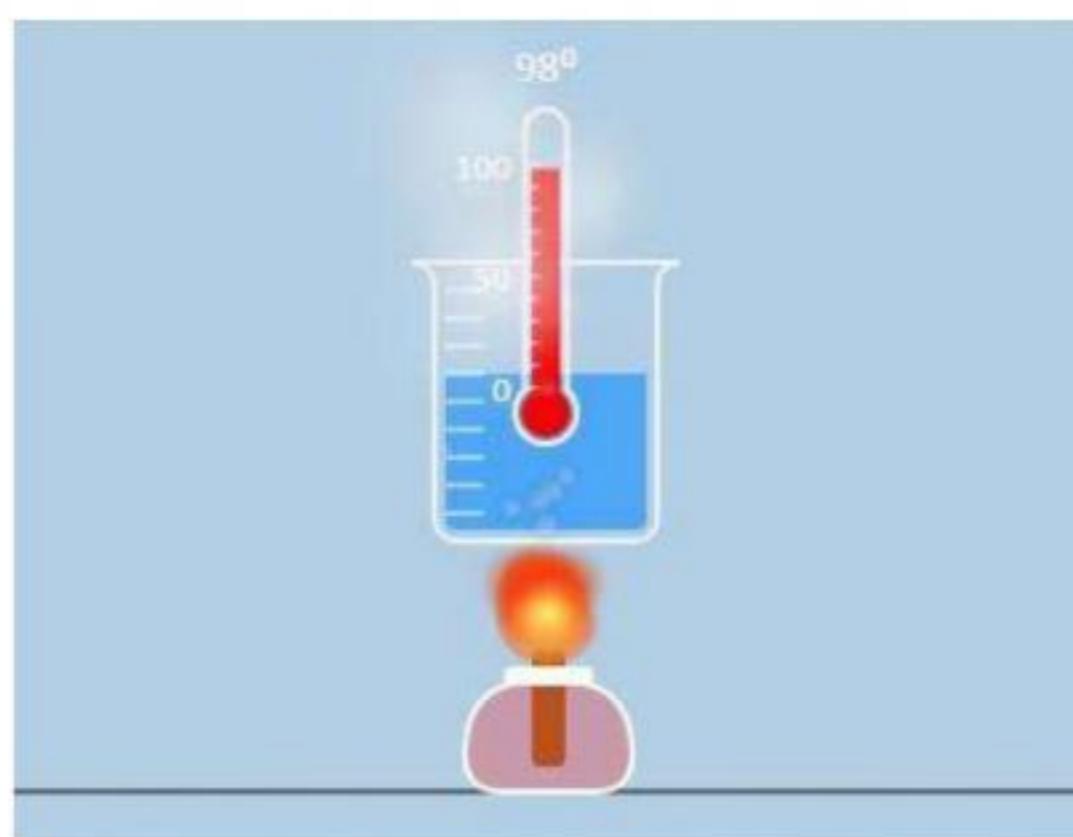
30. termo1.val = suhuK;
31. suhuC = suhuK-273;
32. termo2.val = suhuC;
33. suhuF = (suhuK-273)*1.8+32;
34. termo3.val = suhuF;
```

Selanjutnya untuk menampilkan nilai suhu tersebut pada termometer, digunakan fungsi `termometer` (baris 35 – 37), yang memiliki struktur data sebagai berikut :

|                         |                                                           |
|-------------------------|-----------------------------------------------------------|
| <code>x</code>          | Kordinat x ujung atas termometer                          |
| <code>y</code>          | Kordinat y ujung atas termometer                          |
| <code>l</code>          | Lebar termometer dalam satuan <i>pixel</i>                |
| <code>t</code>          | Tinggi termometer dalam satuan <i>pixel</i>               |
| <code>min</code>        | Skala suhu minimal                                        |
| <code>max</code>        | Skala suhu maksimal                                       |
| <code>val</code>        | Nilai aktif termometer                                    |
| <code>offset</code>     | Jarak antara skala maksimal dengan ujung atas termometer  |
| <code>warnaGaris</code> | Warna garis luar termometer                               |
| <code>warnalsi</code>   | Warna cairan termometer                                   |
| <code>label</code>      | Skala yang ditampilkan pada termometer dalam format array |

## 11.2 Simulasi Pengukuran Panas

Setelah memahami proses simulasi termometer dan konversi suhu, pada tutorial selanjutnya kita akan mensimulasikan proses pemanasan air dalam gelas *beaker*. Pada simulasi ini masing-masing objek dibuat dan dapat digunakan secara interaktif. Adapun rancangan dari simulasi ini adalah sebagai berikut :



Gambar 11.2 Rancangan simulasi pengukuran panas

Pada rancangan simulasi tersebut, pengguna dapat memindahkan termometer ke berbagai arah, dapat menggeser pemanas mematikan dan menyalakannya.

### B. Tutorial 17 – Pengukuran Panas

#### a. *File HTML*

1. Seperti pada projek tutorial sebelumnya, buatlah sebuah *folder* khusus dengan nama **tutorial-17**. *Copy* file **tutorial-16.html** dan *paste* ke dalam *folder*.
2. *Rename* nama *file* menjadi **tutorial-17.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
12. <script src="simulasi-17.js"></script>
```

3. Simpan kembali *file* HTML.

#### b. *File library*

1. *Copy paste* file **simulasiDasar.js** ke dalam *folder* **tutorial-17**.

#### c. *File simulasi-17.js*

1. Pada Notepad++, buatlah sebuah *file* baru.

2. Ketikkan kode berikut :

```
1. aturCanvas();
2. setJudul("Percobaan Pengukuran Suhu");
3. hapusLayar("#b3cfe5");
4.
5. //listener untuk membaca event mouse
6. canvas.onmousedown = mouseDown;
7. canvas.onmouseup = mouseUp;
8.
9. var gelasPiala1 = {tipe:"beaker", x:360, y:250, l:80,
 t:100, rad:8, vol:60, warnaGaris:"#ffffff",
 warnaIsi:"#2495ff"}
10. var apil = {x:200, y:300, v:1, rad:20, max:40,
 warna:"red", blok:gelasPiala1, dist:100};
11. var burner1 = {x:270, y:400, l:60, t:40,
 warnaGaris:"#fff", warnaIsi:"#ff2429"}
12. var termol = {x:100, y:100, l:15, t:80, min:0, max:100,
 val:20, offset: 10, warnaGaris: "#f8f8f8", warnaIsi:
 "red",label:["0", "50", "100"], showVal:true, desimal:0}
13.
14. var apiAktif = 0;
15. var simAktif = 1;
16. var suhuAir = 20;
17. var menguap = false;
18.
19. var dragTermo = {nama:"termo", x:termol.x-25, y:termol.y,
 w:termol.l+50, h:termol.t+40, limit:"xy"};
20. var dragBurner = {nama:"burner", x:burner1.x,
 y:burner1.y, w:burner1.l, h:burner1.t, limit:"x"};
21.
22. function setSimulasi() {
23. hapusLayar();
24. //menampilkan teks
25. teks("Pengukuran Suhu", 0.5*(canvas.width), 40, 'bold
 18pt Calibri', 'blue', 'center');
26. teks("Drag Burner dan termometer untuk mengetahui
 perubahan suhu", 0.5*(canvas.width), 60, "12pt
 Calibri", "#000", "center");
27. //tombol control
28. if (apiAktif == 1){
29. tombol("Matikan/id=api", 360,560, 80, 30, "bold 11pt
 Calibri", "white", "black", "gray", "r");
30. }else{
31. tombol("Nyalakan/id=api", 360,560, 80, 30, "bold 11pt
 Calibri", "white", "black", "gray", "r");
32. }
33. //lantai
34. garis(0,burner1.y+burner1.t, canvas.width,
 burner1.y+burner1.t);
35.
36. //simulasi
37. burner(burner1);
38. tabung(gelasPiala1);
39. if (apiAktif == 1) setApi(burner1, apil);
```

```

40. termometer(termol);
41.
42. //perhitungan api berdasarkan jarak
43. var jarakTermoApi = jarak(termol.x, termol.y+termol.t,
44. apil.x, apil.y);
45. var jarakBeakerApi = jarak(gelasPialal.x+
46. gelasPialal.1/2, gelasPialal.y+gelasPialal.t,
47. apil.x, apil.y);
48. if (apiAktif == 1){
49. //suhu termometer naik
50. if (jarakTermoApi < apil.dist){
51. termol.val+= (apil.dist - jarakTermoApi)/100;
52. }
53. //suhu air naik
54. if (jarakBeakerApi < apil.dist && suhuAir < 100){
55. suhuAir+= (apil.dist - jarakBeakerApi)/1000;
56. }
57. //termometer masuk ke air
58. if (cekHit(termol.ujungX, termol.ujungY, gelasPialal,
59. "vol")){
60. termol.val += (suhuAir - termol.val)/100;
61. }
62. //suhu termometer turun ke suhu normal (misal 20C)
63. if (jarakTermoApi > apil.dist || apiAktif == 0){
64. if (termol.val > 20) termol.val -= 0.025;
65. }
66. //suhu air turun
67. if (jarakBeakerApi > apil.dist || apiAktif == 0) {
68. if (suhuAir > 20) suhuAir -= 0.005;
69. if (suhuAir < 50) menguap = false;
70. }
71. //titik didih air
72. if (suhuAir >= 100){
73. gelembung(gelasPialal);
74. menguap = true;
75. }
76.
77. function jalankanSimulasi() {
78. setSimulasi();
79. if (simAktif == 1) {
80. timer = window.setTimeout(jalankanSimulasi, 10);
81. }
82. }
83.
84. function mouseDown(event){
85. canvas.onmousemove = mouseDrag;
86. startDrag(mouseDrag);
87. }

```

```

88.
89. function mouseDrag(event){
90. //prosedur mengecek objek drag
91. var drag = cekDrag(event);
92. if (drag != null){
93. if (drag.nama == "termo"){
94. termol.x = dragTermo.x+dragTermo.w/2;
95. termol.y = dragTermo.y;
96. }
97. if (drag.nama == "burner"){
98. burner1.x = dragBurner.x;
99. }
100. }
101. }
102.
103. function mouseUp(event){
104. canvas.onmousemove = null;
105. var tombolAktif = cekTombol(event);
106. if (tombolAktif != ""){
107. if (tombolAktif == "api"){
108. if (apiAktif == 1) {
109. apiAktif = 0;
110. }else{
111. apiAktif = 1;
112. }
113. }
114. }
115. }
116.
117. setDrag(dragTermo);
118. setDrag(dragBurner);
119. jalankanSimulasi();

```

3. Simpan dengan nama **simulasi-17.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-17.

Jalankan *file* **tutorial-17.html**, dengan cara membukanya di internet *browser*. Maka kode di atas akan menghasilkan simulasi pengukuran suhu dengan termometer *bulb*, berikut pemanas dan air dalam gelas *beaker* seperti pada gambar 11.3. Pengguna dapat menggerakkan pemanas dan termometer, menyalakan pemanas, mendekatkan termometer ke pemanas atau ke dalam air untuk mengetahui perubahan suhu, dan mengetahui perubahan air ketika terkena panas.



Gambar 11.3 Hasil tutorial pengukuran suhu

#### d. Penjelasan Program

Simulasi pada tutorial 17 tersebut terdapat beberapa objek yang dibuat untuk keperluan simulasi, yaitu termometer yang didefinisikan melalui variabel `termo1` (baris 12), penjelasan tentang struktur data termometer terdapat pada sub bab sebelumnya, namun pada tutorial ini diberikan parameter tambahan yaitu `showVal` dan `desimal` yang berfungsi untuk menampilkan suhu pada bagian atas termometer.

Selanjutnya dibuat sebuah tabung kimia bertipe *beaker* yang didefinisikan melalui variabel `gelasPiala1` (baris 9). Adapun struktur data untuk membentuk tabung kimia tersebut adalah sebagai berikut :

|            |                                                                    |
|------------|--------------------------------------------------------------------|
| tipe       | Tipe gelas kimia dalam hal ini diisi dengan kata “ <i>beaker</i> ” |
| x          | Kordinat x ujung atas tabung kimia                                 |
| y          | Kordinat y ujung atas tabung kimia                                 |
| l          | Lebar tabung kimia dalam satuan <i>pixel</i>                       |
| t          | Tinggi tabung kimia dalam satuan <i>pixel</i>                      |
| rad        | Jari-jari lengkung pada masing-masing sisi tabung kimia            |
| vol        | Volume isi tabung kimia dalam skala prosentase                     |
| warnaGaris | Warna garis luar tabung kimia                                      |
| warnalsi   | Warna cairan di dalam tabung kimia                                 |

Selanjutnya dibuat sebuah pembakar spiritus (*burner*) yang didefinisikan melalui variabel `burner1` (baris 11). Adapun struktur data untuk membentuk *burner* tersebut adalah sebagai berikut :

|            |                                         |
|------------|-----------------------------------------|
| x          | Kordinat x ujung atas <i>burner</i>     |
| y          | Kordinat y ujung atas <i>burner</i>     |
| l          | Lebar <i>burner</i> dalam satuan pixel  |
| t          | Tinggi <i>burner</i> dalam satuan pixel |
| warnaGaris | Warna garis luar <i>burner</i>          |
| warnalsi   | Warna cairan di dalam <i>burner</i>     |

Selanjutnya dibuat variabel untuk menampilkan partikel api yang didefinisikan melalui variabel `api1` (baris 10). Adapun struktur data untuk membentuk api tersebut adalah sebagai berikut :

|       |                                                          |
|-------|----------------------------------------------------------|
| x     | Kordinat x awal munculnya partikel api                   |
| y     | Kordinat y awal munculnya partikel api                   |
| v     | Kecepatan partikel api                                   |
| rad   | Diameter awal kemunculan partikel api dalam satuan pixel |
| max   | Waktu maksimal kemunculan partikel api                   |
| warna | Warna api dapat diisi dengan "red" atau "blue"           |
| blok  | Objek yang dapat menghalangi partikel api                |
| dist  | Jarak maksimal radiasi panas yang dipancarkan oleh api   |

Pada tahapan selanjutnya, dibuatlah sebuah tombol dinamis yang digunakan untuk menyalakan atau mematikan api (baris 28 – 32). Perhatikan bahwa label dari tombol tersebut menyesuaikan kondisi yang ditentukan oleh variabel `apiAktif`. Ketika `apiAktif` bernilai 1 (menyala), maka label yang muncul adalah "matikan", dan sebaliknya ketika `apiAktif` bernilai 0 (padam), maka label yang muncul adalah "nyalakan". Agar mudah dalam mendeteksi tombol tersebut maka ditambahkan akhiran "/id=api", sehingga pengecekan tombol (baris 107), nama tombol yang digunakan adalah "api".

```
28. if (apiAktif == 1){
29. tombol("Matikan/id=api", 360,560, 80, 30, "bold 11pt
 Calibri", "white", "black", "gray", "r");
30. }else{
31. tombol("Nyalakan/id=api", 360,560, 80, 30, "bold 11pt
 Calibri", "white", "black", "gray", "r");
32. }
```

Agar objek *burner* dan termometer dapat dipindah posisinya dengan cara *drag*, maka perlu dilakukan pengaturan objek *drag* melalui variabel `dragTermo` dan `dragBurner`, serta dilakukan aktivasi *drag* dengan kode `setDrag` (baris 117 –

118). Perlu diperhatikan bahwa pada variabel dragTermo parameter `w` (lebar area `drag`), dibuat lebih lebar dari lebar termometer, agar lebih mudah dioperasikan.

Pada tahapan selanjutnya, untuk melakukan simulasi perubahan suhu, dilakukan perhitungan jarak antar api dengan *beaker* (`jarakBeakerApi`) dan antara api dan termometer (`jarakTermoApi`). Ketika jarak tersebut masuk ke area radiasi api (`api1.dist`), maka nilai panas akan dinaikkan, dan sebaliknya panas akan turun perlahan jika jauh dari api atau api sedang dalam kondisi mati.

Sedangkan untuk mengecek suhu air, maka digunakan deteksi antara termometer dengan air dalam *beaker*, dengan metode `cekHit`.

```
56. cekHit(termo1.ujungX, termo1.ujungY, gelasPiala1,"vol")
```

Perhatikan fungsi `cekHit` di atas. Pada fungsi tersebut, dilakukan deteksi antara ujung x dan y objek `termo1` dengan `gelasPiala1`. Diikuti dengan string “`vol`”, karena kita mendeteksi isi (volume) dari `gelasPiala1`, sehingga fungsi `cekHit` tersebut dapat mendeteksi posisi dengan lebih khusus ke isi dari `gelasPiala1`. Tanpa parameter “`vol`”, deteksi akan dilakukan secara menyeluruh antara ujung termometer dengan keseluruhan bidang `gelasPiala1`.

Untuk menunjukkan efek mendidih dan menguap digunakan kode `gelembung(gelasPiala1)` dan `uap(gelasPiala1)`. Fungsi tersebut secara otomatis akan menampilkan efek visual gelembung dan uap pada objek yang dijadikan acuan data.



## BAB 12

### Simulasi Tambahan

#### 12.1 Simulasi Lensa

Lensa merupakan benda bening yang dibatasi oleh dua bidang yang berupa bidang lengkung yang memiliki kemampuan untuk membiaskan cahaya. Lensa dibedakan menjadi dua yaitu lensa cekung dan lensa cembung. Dengan menggunakan simulasi lensa, kita dapat mengatur secara dinamis jenis lensa, fokus lensa, dan posisi objek. Simulasi akan menghitung arah datangnya cahaya dan menentukan posisi serta tinggi bayangan. Untuk membuat simulasi tersebut, perhatikan langkah berikut :

#### A. Tutorial 18 – Simulasi Lensa

##### a. File HTML

1. Seperti pada projek tutorial sebelumnya, buatlah sebuah *folder* khusus dengan nama **tutorial-18**. *Copy file tutorial-17.html* dan *paste* ke dalam *folder*.
2. *Rename* nama *file* menjadi **tutorial-18.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
12. <script src="simulasi-18.js"></script>
```

3. Simpan kembali *file* HTML.

##### b. File library

1. *Copy paste* *file* **simulasiDasar.js** ke dalam *folder* **tutorial-18**.

##### c. File simulasi-18.js

1. Pada Notepad++, buatlah sebuah *file* baru
2. Ketikan kode berikut :

```
1. aturCanvas();
2. setJudul("Lensa");
3. hapusLayar("#b3cfe5");
4.
5. //listener untuk membaca event mouse
6. canvas.onmousedown = mouseDown;
7. canvas.onmouseup = mouseUp;
8.
```

```

9. var x1 = 310;
10. var y1 = 210;
11. var radius = 150;
12. var jarakBenda = -100;
13. var tinggiBenda = 20;
14. var warnaBenda = '#933';
15. var warnaBayangan = '#339';
16. var f = 50;
17. var jarakBayangan;
18. var tinggiBayangan;
19. var slope;
20. var jenisLensa = 1;
21. var lensRadius = 100;
22. var label = "Lensa Cembung";
23.
24. var slider1 = {tipe:"H", nama:"jarak", x:150,y:440,
 p:200, minS:-200, maxS:-25, vals:-100, desimal:0,
 label:"cm"}
25. var slider2 = {tipe:"H", nama:"tinggi", x:150,y:500,
 p:200, minS:-50, maxS:50, vals:20, desimal:0, label:"cm"}
26. var slider3 = {tipe:"H", nama:"fokus", x:150,y:560,
 p:200, minS:30, maxS:100, vals:50, desimal:0, label:"cm"}
27.
28. var simAktif = 1;
29.
30. function setSimulasi(){
31. hapusLayar();
32. //menampilkan teks
33. teks(label, 0.5*(canvas.width), 40, 'bold 18pt
 Calibri', 'blue', 'center');
34. teks("Simulasi cahaya pada lensa cembung dan cekung",
 0.5*(canvas.width), 60, "12pt Calibri", "#000",
 "center");
35.
36. //tombol control
37. tombol("Lensa Cembung", 500, 540, 120, 30, "bold 11pt
 Calibri", "white", "black", "gray", "r");
38. tombol("Lensa Cekung", 640, 540, 120, 30, "bold 11pt
 Calibri", "white", "black", "gray", "r");
39.
40. //slider
41. teks("Jarak Objek = "+slider1.valS+" cm", slider1.x -
 20, slider1.y-20, "bold 13pt Calibri", "black",
 "left");
42. slider(slider1);
43. teks("Tinggi Objek = "+slider2.valS+" cm", slider1.x -
 20, slider2.y-20, "bold 13pt Calibri", "black",
 "left");
44. slider(slider2);
45. teks("Fokus Lensa = "+slider3.valS+" cm", slider3.x -
 20, slider3.y-20, "bold 13pt Calibri", "black",
 "left");
46. slider(slider3);
47.

```

```

48. //gambar lensa
49. if (jenisLensa == 1) {
50. lensaCembung(x1, y1, radius, lensRadius, f);
51. } else {
52. lensaCekung(x1, y1, radius, lensRadius, f);
53. }
54.
55. // gambar sumbu utama
56. garis(0, y1,canvas.width, y1, 2, "black");
57.
58. // gambar titik fokus
59. lingkaran(x1+radius-0.29*radius-2*f, y1, 4, 2,
60. "purple", "purple");
60. lingkaran(x1+radius-0.29*radius-4*f, y1, 4, 2,
61. "purple", "none");
61. lingkaran(x1+radius-0.29*radius+2*f, y1, 4, 2,
62. "purple", "purple");
62. lingkaran(x1+radius-0.29*radius+4*f, y1, 4, 2,
63. "purple", "none");
63.
64. jarakBayangan = -jarakBenda*f/(-jarakBenda-f);
65. tinggiBayangan = jarakBayangan*tinggiBenda/(jarakBenda);
66.
67. garis(x1+radius-0.29*radius+2*jarakBenda, y1-
68. 2*tinggiBenda, x1+radius-0.29*radius, y1-2*tinggiBenda,
69. 2, "red");
70. slope = tinggiBenda/f;
71. garis(x1+radius-0.29*radius, y1-2*tinggiBenda,
72. x1+radius-0.29*radius+420, y1-2*tinggiBenda+slope*420,
73. 2, "red");
73.
74. slope = -tinggiBenda/jarakBenda;
75. garis(x1+radius-0.29*radius+2*jarakBenda,
76. y1-2*tinggiBenda, x1+radius-0.29*radius+420,
77. y1-2*tinggiBenda+slope*(420-2*jarakBenda), 2, "blue");
76.
77. if (jarakBayangan < 0.0) {
78. garis(x1+radius-0.29*radius+2*jarakBenda,
79. y1-2*tinggiBenda, x1+radius-0.29*radius-420,
80. y1-2*tinggiBenda-slope*(420+2*jarakBenda, 2,
81. "#bbf"));
81.
82. // gambar garis cahaya
83. slope = tinggiBenda/(jarakBenda+f);
84. garis(x1+radius-0.29*radius+2*jarakBenda,
85. y1-2*tinggiBenda, x1+radius-0.29*radius,
86. y1-2*tinggiBenda+slope*(2*jarakBenda), 2, "green");
87. garis(x1+radius-0.29*radius, y1-2*tinggiBenda+
88. slope*(2*jarakBenda), x1+radius-0.29*radius+420,

```

```

 y1-2*tinggiBenda+slope*(2*jarakBenda), 2, "green");
85.
86. if (jarakBayangan < 0.0) {
87. garis(x1+radius-0.29*radius, y1-2*tinggiBenda+
88. slope*(2*jarakBenda),x1+radius-0.29*radius-420,
89. y1-2*tinggiBenda+slope*(2*jarakBenda), 2, "#bf8");
90. }
91. // objek
92. if (tinggiBenda > 0) panah(x1+radius-0.29*radius+
93. 2*jarakBenda,y1,0,tinggiBenda, 2, 2, warnaBenda);
94. if (tinggiBenda < 0) panah(x1+radius-0.29*radius+
95. 2*jarakBenda,y1,0,tinggiBenda, 2, 2, warnaBenda);
96. // bayangan
97. if (tinggiBayangan > 0) panah(x1+radius-0.29*radius+
98. 2*jarakBayangan,y1,0,tinggiBayangan, 2, 2,
99. warnaBayangan);
100. if (tinggiBayangan < 0) panah(x1+radius-0.29*radius+
101. 2*jarakBayangan,y1,0,tinggiBayangan, 2, 2,
102. warnaBayangan);
103. teks("Jarak Bayangan = "+jarakBayangan.toFixed(2)+" cm",
104. 500, 420, "bold 13pt Calibri", "black", "left");
105. teks("Tinggi Bayangan = "+tinggiBayangan.toFixed(2)+" cm",
106. 500, 450, "bold 13pt Calibri", "black", "left");
107. }
108. function mouseDown(event){
109. canvas.onmousemove = mouseDrag;
110. }
111. function mouseDrag(event){
112. //prosedur mengecek slider
113. var sliderAktif = cekSlider(event);
114. if (sliderAktif != null){
115. if (sliderAktif.nama == "jarak") {
116. jarakBenda = Number(sliderAktif.vals);
117. setSimulasi();
118. }
119. if (sliderAktif.nama == "tinggi") {
120. tinggiBenda = Number(sliderAktif.vals);
121. setSimulasi();
122. }
123. }
124. }
125. function mouseUp(event){

```

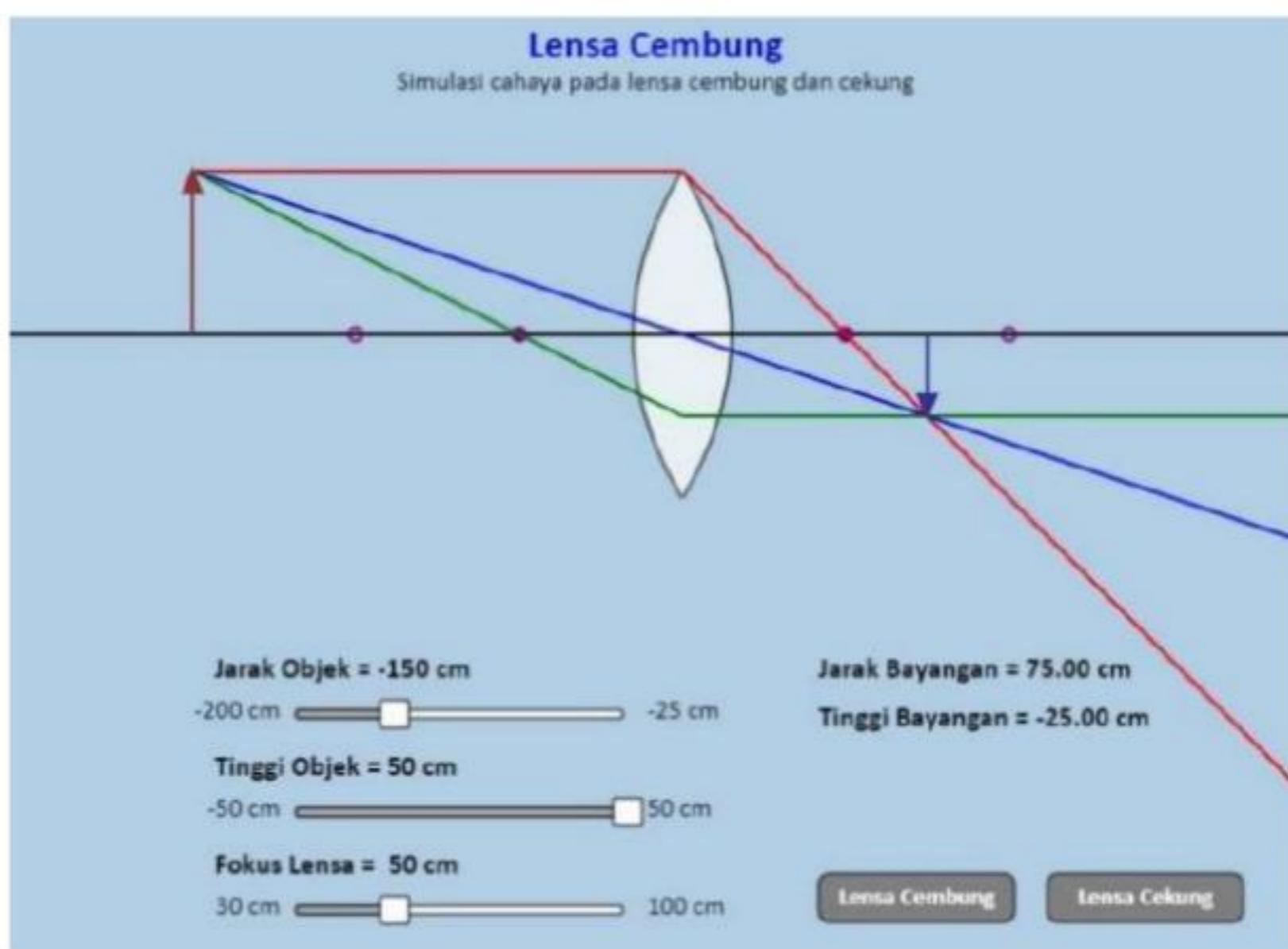
```

127. //prosedure mengecek tombol
128. var tombolAktif = cekTombol(event);
129. if (tombolAktif != "") {
130. if (tombolAktif == "Lensa Cembung") {
131. f=Math.abs(f);
132. jenisLensa = 1;
133. label = "Lensa Cembung";
134. setSimulasi();
135. }
136. if (tombolAktif == "Lensa Cekung") {
137. f=-1*Math.abs(f);
138. jenisLensa = 2;
139. label = "Lensa Cekung";
140. setSimulasi();
141. }
142. }
143. canvas.onmousemove = null;
144. }
145.
146. setSimulasi();

```

3. Simpan dengan nama **simulasi-18.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-18.

Jalankan file **tutorial-18.html**, dengan cara membukanya di *internet browser*. Maka kode di atas akan menghasilkan simulasi lensa cembung dan cekung dengan beberapa variabel yang dapat diatur secara dinamis sebagai berikut :



Gambar 12.1 Hasil tutorial lensa

#### d. Penjelasan Program

Untuk membentuk simulasi lensa, dilakukan perhitungan sederhana untuk menghasilkan jarak bayangan dan tinggi bayangan, lihat baris 64-65.

```
64. jarakBayangan = -jarakBenda*f/(-jarakBenda-f);
65. tinggiBayangan = jarakBayangan*tinggiBenda/(jarakBenda);
```

Dengan ditemukannya jarak bayangan dan tinggi bayangan kita dapat menarik garis-garis cahaya dengan menggunakan fungsi `garis` dan `panah`.

## 12.2 Simulasi Medan Listrik

Medan listrik merupakan medan gaya yang ditimbulkan oleh keberadaan muatan listrik, seperti elektron, ion, atau proton, dalam ruangan yang ada di sekitarnya. Medan listrik termasuk dalam vektor, arah medan listrik dinyatakan sama dengan arah gaya yang dialami benda muatan.

Pada tutorial berikutnya, akan dibuat sebuah simulasi yang menggambarkan garis medan listrik. Pengguna dapat merubah nilai muatan listrik antara 2 titik dan arah garis medan listrik akan berubah secara dinamis sesuai dengan nilai muatan yang terdefinisikan. Untuk membuat simulasi tersebut, perhatikan langkah berikut :

### A. Tutorial 19 – Simulasi Medan Listrik

#### a. File HTML

1. Seperti pada projek tutorial sebelumnya, buatlah sebuah *folder* khusus dengan nama **tutorial-19**. *Copy file tutorial-18.html* dan *paste* ke dalam *folder*.
2. *Rename* nama *file* menjadi **tutorial-19.html**. Kemudian ubah baris 12, dan sesuaikan dengan nomor tutorial, menjadi sebagai berikut :

```
12. <script src="simulasi-19.js"></script>
```

3. Simpan kembali *file* HTML.

#### b. File library

1. *Copy paste* *file* **simulasiDasar.js** ke dalam *folder* **tutorial-19**.

### c. File simulasi-19.js

1. Pada Notepad++, buatlah sebuah *file* baru

2. Ketikan kode berikut :

```
1. aturCanvas();
2. setJudul("Medan Listrik");
3. hapusLayar("#1b53a8");
4.
5. //listener untuk membaca event mouse
6. canvas.onmousedown = mouseDown;
7. canvas.onmouseup = mouseUp;
8.
9. var index = 0;
10. var xBase = 50;
11. var yBase = 300;
12. var radius = 10;
13. var charge1X = xBase+200;
14. var charge1Y = yBase;
15. var charge2X = xBase+500;
16. var charge2Y = yBase;
17. var Ex = 0;
18. var Ey = 0;
19. var rSq = 0;
20. var xPos = 0;
21. var yPos = 0;
22. var vectorColor = 'black';
23. var Q1 = 5;
24. var Q2 = -5;
25.
26. var slider1 = {tipe:"H", nama:"Muatan 1", x:100,y:540,
 p:200, minS:-10, maxS:10, vals:Q1, desimal:0, label:"C"}
27. var slider2 = {tipe:"H", nama:"Muatan 2", x:450,y:540,
 p:200, minS:-10, maxS:10, vals:Q2, desimal:0, label:"C"}
28.
29. function setSimulasi(){
30. hapusLayar();
31. gridBG();
32. //menampilkan teks
33. teks("Medan Listrik", 0.5*(canvas.width), 40, 'bold
 18pt Calibri', '#d9f4ff', 'center');
34. teks("Simulasi menunjukkan medan listrik di dekat dua
 partikel bermuatan.", 0.5*(canvas.width), 75, "12pt
 Calibri", "#d9f4ff");
35.
36. teks("Muatan A = "+slider1.vals+ " C", slider1.x-20,
 slider1.y - 20, "bold 13pt Calibri", "black", "left");
37. slider(slider1);
38. teks("Muatan B = "+slider2.vals+ " C", slider2.x-20,
 slider2.y - 20, "bold 13pt Calibri", "black", "left");
39. slider(slider2);
40.
41. var w1, w2;
```

```

42. // gambar medan gaya
43. for (var i = -7; i <= 7; i++) {
44. Ex = Ey = 0;
45. xPos = 400+40*i;
46. for (var j = -4; j<= 4; j++) {
47. yPos = yBase + 40*j;
48. rSq = (xPos-charge1X)*(xPos-charge1X) +
49. (yPos-charge1Y)*(yPos-charge1Y);
50. Ex = 6000*Q1*(xPos-charge1X)/Math.pow(rSq,1.5);
51. Ey = 6000*Q1*(charge1Y-yPos)/Math.pow(rSq,1.5);
52. rSq = (xPos-charge2X)*(xPos-charge2X) +
53. (yPos-charge2Y)*(yPos-charge2Y);
54. Ex = Ex - 6000*Q2*(charge2X-xPos)/Math.pow(rSq,1.5);
55. Ey = Ey - 6000*Q2*(yPos-charge2Y)/Math.pow(rSq,1.5);
56. E = Math.sqrt(Ex*Ex + Ey*Ey);
57. Ex = 6*Ex/E;
58. Ey = 6*Ey/E;
59. E = E*40;
60. if (E > 255) E = 255;
61. E = Math.round(255-E);
62. grayString = E.toString(16);
63. if (grayString.length == 1)
64. grayString = "0"+grayString;
65. vectorColor = "#" + grayString+ grayString+
66. grayString;
67. panah(xPos,yPos,Ex,Ey, 5, 3, vectorColor);
68. }
69. }
70. //simulasi medan listrik
71. if (Math.abs(Q1) > 0.2) {
72. if (Q1 > 0) {
73. w1 = "red";
74. } else {
75. w1 = "blue";
76. }
77. lingkaran(charge1X, charge1Y, radius, 2, "white", w1);
78. }
79. //simulasi medan listrik
80. if (Math.abs(Q2) > 0.2) {
81. if (Q2 > 0) {
82. w2 = "red";
83. } else {
84. w2 = "blue";
85. }
86. lingkaran(charge2X, charge2Y, radius, 2, "white", w2);
87. }
88. function mouseDown(event){
89. canvas.onmousemove = mouseDrag;

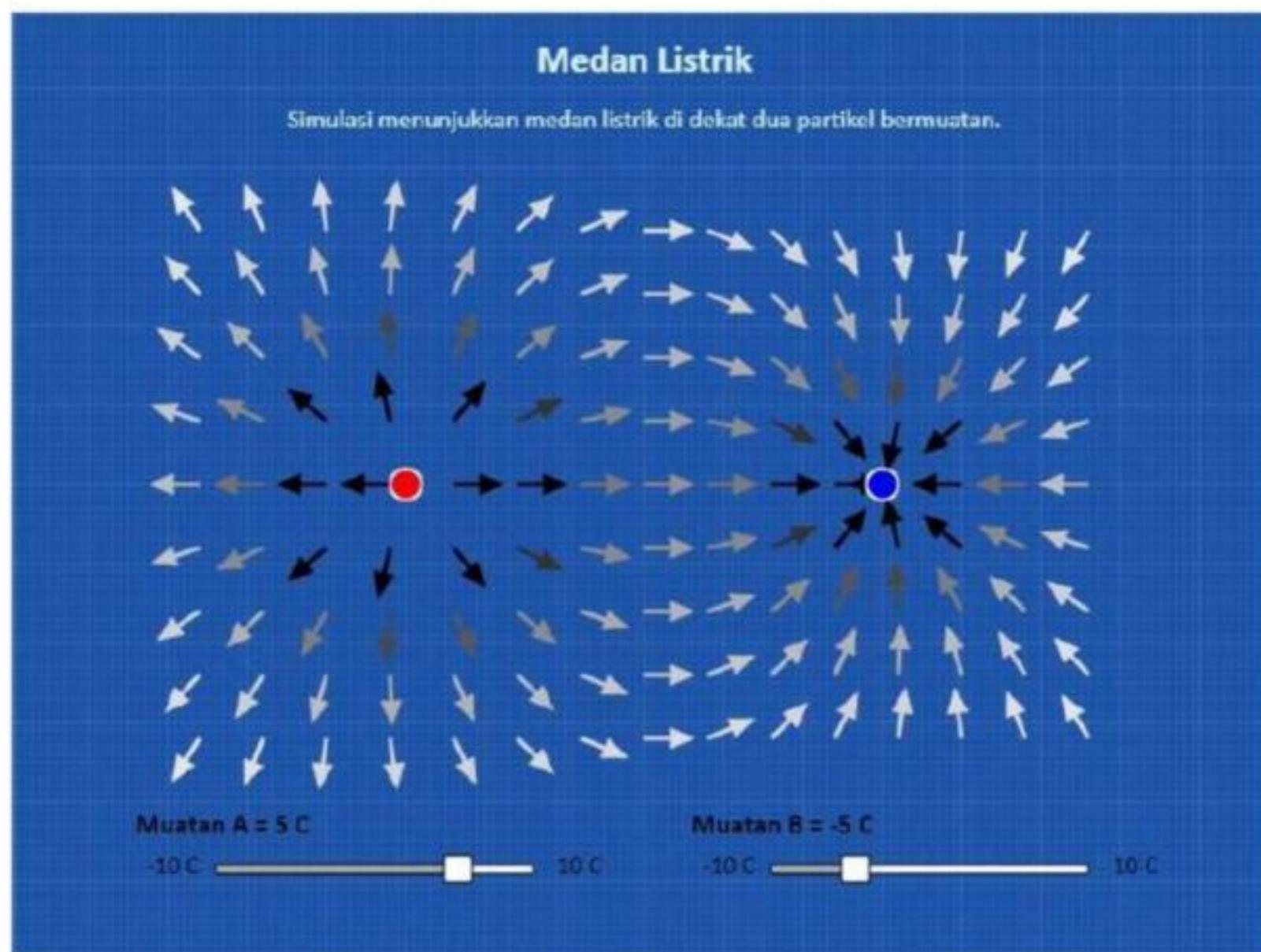
```

```
90. function mouseUp(event) {
91. //menetralisir drag
92. canvas.onmousemove = null;
93. }
94.
95. function mouseDrag(event) {
96. //prosedur mengecek slider
97. var sliderAktif = cekSlider(event);
98. if (sliderAktif != null){
99. if (sliderAktif.nama == "Muatan 1") {
100. Q1 = sliderAktif.valS;
101. setSimulasi();
102. }
103. if (sliderAktif.nama == "Muatan 2") {
104. Q2 = sliderAktif.valS;
105. setSimulasi();
106. }
107. }
108. }
109.
110. setSimulasi();
```

3. Simpan dengan nama **simulasi-19.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-19.

Jalankan *file* **tutorial-19.html**, dengan cara membukanya di internet *browser*. Maka kode di atas akan menghasilkan simulasi medan listrik seperti pada gambar 12.2. Perhatikan perubahan arah vektor ketika muatan listrik ke dua titik diubah dengan mengeser *slider*. Pada simulasi sederhana ini kita dapat melihat hubungan tarik-menarik atau saling menolak antara 2 muatan ketika nilai muatan berubah. Vektor panah secara otomatis menyesuaikan arah dan warnanya besaran muatan.

Pada dasarnya dengan teknik yang sama juga dapat dibuat simulasi medan magnet untuk menunjukkan hubungan antara kutub-kutub magnet.



Gambar 12.2 Hasil tutorial medan listrik

#### d. Penjelasan Program

Untuk membentuk garis medan gaya, dilakukan perhitungan berulang dengan operasi `for` bertingkat, lihat baris 43-66. Pada perhitungan berulang tersebut dihitung posisi panah vektor dengan variabel `xPos`, `yPos` untuk ujung awal panah, dan `Ex`, `Ey` untuk ujung akhir panah. Untuk memberikan warna yang berbeda-beda tergantung kekuatan muatan listrik, digunakan variabel `grayString`.

## BAB 13

### ***GUI (Graphic User Interface)***

#### **13.1 Antar Muka (*GUI*)**

Setelah memahami beberapa teknis pembuatan simulasi, maka tahap berikutnya adalah menyajikan simulasi tersebut kepada pengguna. Pada dasarnya, kita dapat secara langsung menjadikan simulasi yang kita buat tersebut sebagai produk final, namun terdapat beberapa pertimbangan untuk menambahkan beberapa elemen antar muka (*GUI*). Sebagai contoh saat pertama kali aplikasi dijalankan, pengguna akan mendapati halaman judul terlebih dahulu, halaman petunjuk operasional, halaman kompetensi yang diharapkan, barulah pengguna dapat melakukan praktikum.

Dalam menyusun antar muka sebuah aplikasi diperlukan seperangkat aset visual yang meliputi grafik yang akan dipakai sebagai objek dalam media, efek visual, dan typografi (penggunaan huruf). Aset visual yang dimaksud dapat dikembangkan dengan menggunakan *software* grafis seperti Adobe Photoshop, Adobe Ilustrator atau aplikasi grafis lainnya, dan dapat disimpan dalam format *bitmap* (JPG atau PNG). Ketika seorang pengembang aplikasi kurang menguasai teknis grafis, maka dapat memanfaatkan aset visual yang terdapat secara *online* (pada website tertentu), dengan memperhatikan lisensi dari aset visual yang dimaksud.

Antar muka pertama dalam sebuah aplikasi adalah halaman pembuka (halaman judul). Halaman pembuka secara umum tersusun atas *background* ilustrasi, judul aplikasi, dan tombol untuk memulai aplikasi. Halaman pembuka pada aplikasi memiliki fungsi sebagai *first impression* sebuah aplikasi yang berarti halaman yang pertama kali dilihat oleh pengguna, sehingga halaman ini harus memiliki daya tarik, yang bisa diwujudkan melalui ilustrasi gambar, warna dan tata letak yang baik. Sebagai contoh dalam aplikasi yang dibuat di buku ini digunakan judul *Virtual Lab*, dengan tipografi (huruf) yang mengesankan tema laboratorium (dengan penambahan ilustrasi gelas kimia, magnet dan roda gigi). Tampilan judul ini disimpan dalam *file* bertipe PNG untuk mendapatkan latar belakang (*background*) transparan.



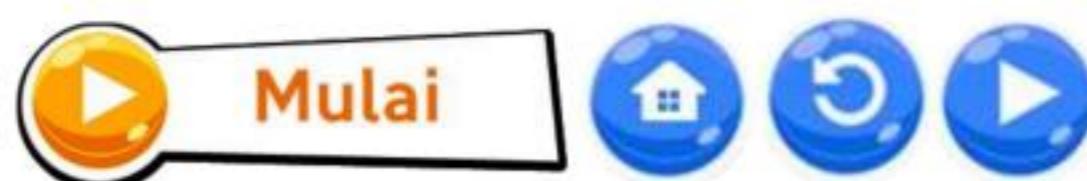
Gambar 13.1 Tampilan grafis judul aplikasi

Dalam aplikasi ini juga digunakan sebuah latar belakang (*background*) dengan ilustrasi beberapa peralatan laboratorium, dan diberikan garis-garis untuk mengesankan efek *blueprint*. Hal yang perlu diperhatikan dalam membuat latar belakang ini adalah mengeset ukuran gambar sebagaimana luaran yang diharapkan. Sebagai contoh, ketika mengembangkan aplikasi untuk komputer atau *desktop*, maka dapat menggunakan ukuran 800x600 *pixel*, atau ukuran yang paling umum yaitu 1280 x 720 *pixel*. Sedangkan ketika mengembangkan aplikasi untuk keperluan *mobile phone* (*platform* Android atau iOS), maka digunakan ukuran grafis dengan rasio 2:1, sebagai contoh menggunakan ukuran 1200 x 600 *pixel*.



Gambar 13.2 Grafis untuk latar belakang aplikasi (1200 x 600 *pixel*)

Salah satu elemen visual dari aplikasi adalah seperangkat tombol. Dalam *library* simulasiDasar.as, telah dilengkapi dengan fitur tombol (membuat tombol tanpa aset visual eksternal), namun tombol yang dihasilkan masih tergolong sangat sederhana. Sedangkan dalam sebuah aplikasi yang profesional, kita membutuhkan sebuah tombol yang memiliki tampilan yang menarik. Sebagai contoh pada aplikasi ini digunakan beberapa aset visual untuk tombol dengan *file* bertipe PNG.



Gambar 13.3 Contoh aset visual untuk tombol

Elemen visual berikutnya yang sering kali ditemukan dalam sebuah aplikasi yang ditujukan sebagai media pembelajaran seperti virtual lab adalah agen pedagogis. Agen pedagogis merupakan konsep yang dipinjam dari ilmu komputer dan kecerdasan buatan (*AI*) yang diterapkan untuk pendidikan. Agen pedagogis berupa antarmuka yang biasanya dipresentasikan secara visual mirip dengan manusia dan mensimulasikan hubungan antara pengguna dan konten edukasi, dalam lingkungan pendidikan. Agen pedagogis secara umum divisualisasikan dalam bentuk avatar (perwakilan sebuah karakter), yang menyampaikan pesan-pesan edukasi tertentu, dibuat semenarik mungkin agar dapat berinteraksi dengan pengguna. Berbagai penelitian telah membuktikan bahwa keberadaan agen pedagogis sangat efektif dalam meningkatkan daya serap seorang pengguna aplikasi pembelajaran. Dalam aplikasi ini misalnya, digunakan agen pedagogis dengan karakter yang dibuat mirip dengan karakter Einstein, seorang figur yang identik dengan bidang fisika.



Gambar 13.4 Agen pedagogis

Setelah seluruh aset visual disiapkan, maka tahapan selanjutnya adalah menyusun aset visual tersebut menjadi sebuah antar muka dengan menggunakan kode.

### 13.2 Menyiapkan Simulasi yang Akan Ditampilkan dalam Aplikasi

Dalam sebuah aplikasi laboratorium virtual, tentusaja pengguna dapat melakukan beberapa macam praktikum, tidak hanya satu praktikum seperti yang dibuat pada tutorial-tutorial sebelumnya. Untuk menambahkan beberapa jenis praktikum ke

dalam satu aplikasi diperlukan beberapa persiapan khusus, agar nantinya aplikasi bekerja dengan baik tanpa adanya kode yang tumpang tindih.

Sebagai contoh dalam tutorial yang ada di bab ini, akan digunakan 3 buah simulasi gerak yaitu simulasi gerak dan percepatan (tutorial-9), simulasi gerak parabola (tutorial-11) dan simulasi gerak melingkar (tutorial-12). Pada masing-masing tutorial tersebut terdapat kode Javascript yang identik (memiliki beberapa nama fungsi dan nama variabel yang sama), sehingga apabila kita *load* secara bersama-sama akan terjadi konflik pada kode. Maka kita perlu sedikit memodifikasi kode tersebut. Perhatikan langkah berikut :

#### A. Tutorial 20 - GUI Virtual Lab

##### a. **File simulasi-9.js, simulasi-11 js, dan simulasi-12.js**

1. Buatlah sebuah *folder* baru dengan nama **tutorial-20**. Folder ini nantinya akan kita gunakan untuk menyimpan seluruh *file* untuk tutorial di bab ini.
2. Buka *file* **simulasi-9.js** (file simulasi gerak dan percepatan), dimana *file* tersebut mensimulasikan 2 mobil yang bergerak dan terdapat pada *folder* tutorial-9.
3. *Copy* seluruh kode yang ada pada *file* simulasi-9 tersebut.
4. Buka Notepad++, dan buatlah sebuah *file* baru
5. Ketikkan satu baris kode berikut :

```
1. function simulasi1() {
```

6. Kemudian *pastekan* kode yang kita *copy* pada langkah 2.
7. Pada kode yang kita *pastekan*, terdapat fungsi *jalankanSimulasi()*, yang di dalamnya terdapat kode

```
timer = window.setTimeout(jalankanSimulasi, 20);
```

Dalam praktik penggunaan aplikasi, sangat dimungkinkan seorang pengguna keluar dari simulasi sebelum simulasi tersebut berakhir (keluar pada saat simulasi masih berjalan). Hal ini akan menyebabkan aplikasi mengalami masalah, karena masih ada sebuah perintah yang berjalan (*timer*), sementara kode sudah memerintahkan untuk keluar dari perintah tersebut. Sehingga kita perlu menambahkan sedikit kode untuk mengatasinya.

Pada kode yang kita *paste* kan (langkah 6), carilah baris yang memiliki fungsi `jalankanSimulasi`, kemudian tambahkan satu baris (baris 179) ke dalam fungsi tersebut. Perhatikan detail berikut :

```
175. function jalankanSimulasi() {
176. setSimulasi();
177. if (simAktif == 1) {
178. timer = window.setTimeout(jalankanSimulasi, 20);
179. mainTimer = timer;
180. }
181. }
```

Variabel `mainTimer` ini nantinya akan kita gunakan untuk menghentikan segala simulasi yang aktif (lihat baris 24 dan baris 146 pada *file* simulasi-20.js)

8. Selanjutnya pada baris terakhir (paling bawah), tambahkan karakter kurung kurawal tutup `}`, sehingga keseluruhan kode pada file ini menjadi sebagai berikut:

```
1. function simulasi1(){
2. aturCanvas();
3. setJudul("Gerak dan Kecepatan");
4. . . .
5. . . . dan seterusnya, code yang dipaste dari simulasi-9
201. }
```

9. Simpan dengan nama **simulasi1.js**. Letakkan *file* pada *folder* tutorial-20.
10. Selanjutnya kita akan melakukan hal yang sama untuk tutorial gerak melingkar dan tutorial gerak parabola. Lakukan langkah no 2 – 9, dengan catatan pada langkah no 5, anda sesuaikan kode baris pertamanya menjadi `function simulasi2()` { untuk tutorial-11 dan `function simulasi3()` { untuk tutorial-12.
11. Simpan *file* seperti pada langkah 9, dengan penyesuaian nama, sehingga pada langkah ini kita memiliki 3 buah *file*: **simulasi1.js**, **simulasi2.js** dan **simulasi3.js**.

### b. File HTML

Berbeda dengan tutorial sebelumnya, pada tutorial ini kita membutuhkan beberapa *file* yang harus *diload*, yaitu file **simulasi1.js**, **simulasi2.js**, **simulasi3.js** serta *file* **simulasi-20.js** (yang akan dibuat pada langkah selanjutnya). Untuk membuat *file* HTML yang dimaksud, ikuti langkah berikut :

1. Buatlah sebuah *file* baru di Notepad++, lalu ketikkan kode berikut

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4. <meta charset="utf-8" />
5. <title>Judul Simulasi</title>
6. <script src='simulasiDasar.js'></script>
7. <script src='simulasi1.js'></script>
8. <script src='simulasi2.js'></script>
9. <script src='simulasi3.js'></script>
10. </head>
11. <body>
12. <center>
13. <canvas id="scene" width="1200" height="600" >
14. </canvas>
15. </center>
16. <script src="simulasi-20.js"></script>
17. </body>
18. </html>
```

2. Simpan dengan nama **tutorial-20.html**. Pastikan ekstensi *file* tersebut adalah HTML dan simpan pada *folder* tutorial-20.

Perhatikan pada baris 7-9 di atas, kita membuka *file* yang kita buat pada tahapan sebelumnya (tahap a), sedangkan pada baris 16, kita akan membuka file yang akan kita buat pada tahap berikutnya (tahap d)

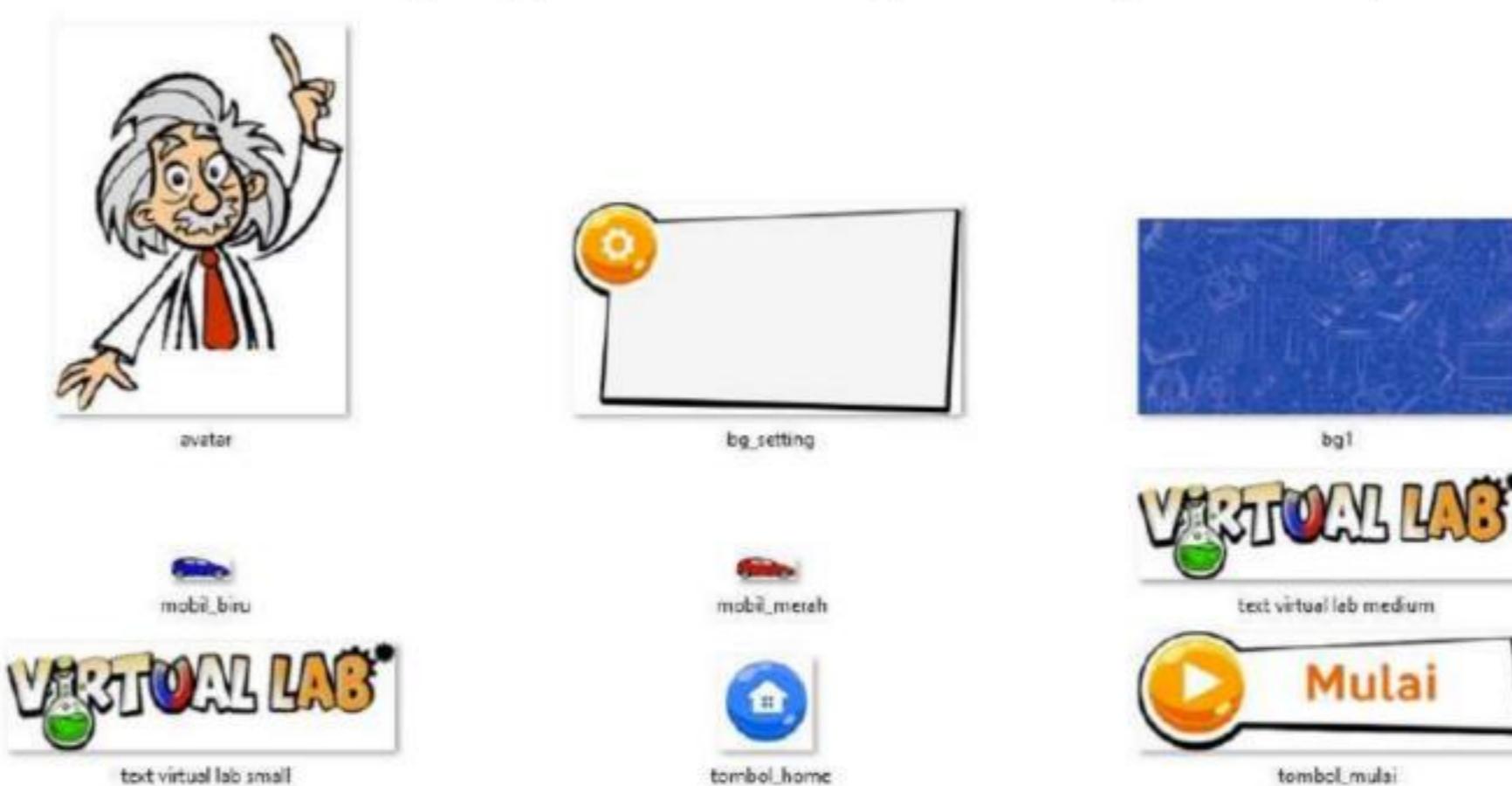
### c. File *library* dan file gambar

1. *Copy paste* *file* **simulasiDasar.js** ke dalam *folder* **tutorial-20**.
2. Selanjutnya buatlah sebuah *folder* dengan nama **images**.
3. Siapkan beberapa gambar yang akan digunakan sebagai elemen antar muka (*GUI*), dan letakkan asset visual tersebut ke dalam *folder* images.

Dalam contoh tutorial ini digunakan gambar sebagai berikut :

- *avatar.png* yang merupakan gambar karakter Einstein

- bg\_setting.png yang merupakan gambar latar untuk menanyakan nama pemain
- bg1.jpg yang merupakan latar belakang aplikasi dengan ukuran 1200x600 pixel
- teks virtual lab medium.png yang merupakan judul aplikasi dengan ukuran sedang
- text virtual lab small.png yang merupakan judul aplikasi dengan ukuran kecil
- tombol\_home.png yang merupakan tombol bulat, untuk keluar dari simulasi
- tombol\_mulai.png yang merupakan tombol untuk memulai aplikasi
- mobil\_biru.png dan mobil\_merah.png yang merupakan bagian dari simulasi-9 (yang pada tutorial ini dijadikan sebagai simulasi1).



Gambar 13.5 Gambar yang digunakan pada tutorial-20

*File-file* gambar tersebut dapat ada buat sendiri, atau anda *copy-paste* dari *file* tutorial yang disertakan di dalam buku. Anda dapat melakukan penyesuaian nama *file* pada kode di simulasi-20.js baris 10-16 apabila anda menggunakan sumber gambar dengan nama yang berbeda.

**d. File simulasi-20.js**

1. Pada Notepad++, buatlah sebuah *file* baru

2. Ketikan kode berikut :

```
1. aturCanvas();
2. setJudul("Virtual Lab");
3. hapusLayar("#1b53a8");
4.
5. //listener untuk membaca event mouse
6. canvas.onmousedown = mouseDown;
7. canvas.onmouseup = mouseUp;
8.
9. var fileGUI = {
10. bg: "images/bg1.jpg",
11. judul: "images/text virtual lab medium.png",
12. juduls: "images/text virtual lab small.png",
13. bg_setting: "images/bg_setting.png",
14. avatar: "images/avatar.png",
15. tombol_home: "images/tombol_home.png",
16. tombol_mulai: "images/tombol_mulai.png"
17. };
18.
19. var inputNama = {nama:"nama", x:520, y:350, p:200, t:30,
20. huruf:"13pt Calibri", val:"nama", max:30, limit:"*",
21. warnaLayar:"#f0f0f0"};
22. var popup1 = {x:400, y:250, l:400, t:150,
23. warnaBG:"#f7f7f7", warnaGaris:"#bababa", val:"",
24. huruf:"14pt-Calibri-center-1.5", warnaHuruf:"#8c1515",
25. tutup:"ok", func: cekNamaPengguna}
26. preload(fileGUI, halamanJudul);
27.
28. function halamanJudul(){
29. hapusLayar();
30. //menampilkan gui untuk judul
31. gambar(dataGambar.bg, 0, 0);
32. gambar(dataGambar.judul, 300, 50);
33. tombolImg("mulai", 400, 300, 337, 103,
34. dataGambar.tombol_mulai);
35.
36. function halamanNama(){
37. hapusLayar();
38. //menampilkan gui untuk judul
39. gambar(dataGambar.bg, 0, 0);
40. kotak(0,0,canvas.width, canvas.height, 1, "none",
41. "rgba(19,41,155,0.7)");
42. gambar(dataGambar.juduls, 300, 150);
```

```

42. gambar(dataGambar.avatar, 600, 50);
43. gambar(dataGambar.bg_setting, 400, 250);
44. //teks html
45. teksHTML("Selamat datang di VIRTUAL LAB,

sebelum memulai tuliskan nama anda terlebih dahulu
 !", 500, 280, 250, "12pt-Arial-center-1.6", "#690608");
46. teksInput(inputNama);
47. //tombol OK
48. tombol("OK/id=cek_nama", 575, 400, 80, 30, "bold 14pt
 Calibri", "white", "#12b098", "#12b098", "r");
49. }
50.
51. function halamanMenu(){
52. hapusLayar("#b3cfe5", {stat:"clear"});
53. gambar(dataGambar.bg, 0, 0);
54. kotak(0,0,canvas.width, canvas.height, 1, "none",
 "rgba(19,41,155,0.7)");
55. gambar(dataGambar.juduls, 450, 50);
56.
57. //petunjuk
58. kotakrs(300, 480, 600, 100, 10, 2, "#8f8f8f",
 "#e6e6e6", "black");
59. gambar(dataGambar.avatar, 150, 380);
60. teksHTML("Haiii... "+namaPengguna+" silahkan
 Klik Simulasi Fisika yang anda inginkan
Klik
 tombol kembali jika anda ingin memilih ulang jenis
 simulasi!", 350, 500, 500, "12pt-Arial-center-1.6",
 "#690608");
61.
62. //tombol menu simulasi
63. tombol("Gerak dan Kecepatan/id=simulasi1", 450, 200,
 300, 40, "bold 14pt Calibri", "white", "#d49715",
 "#d49715", "r");
64. tombol("Gerak Parabola/id=simulasi2", 450, 250, 300,
 40, "bold 14pt Calibri", "white", "#d49715", "#d49715",
 "r");
65. tombol("Gerak Melingkar/id=simulasi3", 450, 300, 300,
 40, "bold 14pt Calibri", "white", "#d49715", "#d49715",
 "r");
66. }
67.
68. function tambahHome(){
69. //tombol kembali
70. gambar(dataGambar.juduls, 880, 10);
71. tombolImg("home", 10, 10, 60, 60,
 dataGambar.tombol_home, keluarSimulasi);
72. }
73.
74. function halamanSimulasi(){
75. //hapus layar dan tambahkan tombol home
76. hapusLayar("#b3cfe5", {stat:"run", func:tambahHome});
77. //jalankan simulasi dari file yang diload diawal file
78. if (noSimulasi == 1) simulasi1();
79. if (noSimulasi == 2) simulasi2();

```

```

80. if (noSimulasi == 3) simulasi3();
81. }
82.
83. function cekNamaPengguna(){
84. if (namaPengguna == "") {
85. halamanNama();
86. }else{
87. resetInteraktif();
88. halamanMenu();
89. }
90. }
91.
92. function mouseDown(event){
93. canvas.onmousemove = mouseDrag;
94. }
95.
96. function mouseDrag(event){
97. }
98.
99. function mouseUp(event){
100. cekTeksInput(event);
101. cekPopup(event);
102. //prosedure mengecek tombol
103. var tombolAktif = cekTombol(event);
104. if (tombolAktif != ""){
105. if (tombolAktif == "mulai"){
106. resetInteraktif();
107. halamanNama();
108. }
109. //mengecek nama pemain
110. if (tombolAktif == "cek_nama"){
111. //nama belum diinputkan
112. if (inputNama.val == "nama" ||
113. inputNama.val == ""){
114. popup1.val = "Anda belum mengetikan nama,
115. silahkan klik pada kolom nama dan ketikan
116. nama anda!";
117. }
118. }
119. popup(popup1);
120. }
121. if (tombolAktif == "simulasil"){
122. noSimulasi = 1;
123. resetInteraktif();
124. halamanSimulasi();
125. }
126. if (tombolAktif == "simulasi2") {

```

```

127. noSimulasi = 2;
128. resetInteraktif();
129. halamanSimulasi();
130. }
131. if (tombolAktif == "simulasi3"){
132. noSimulasi = 3;
133. resetInteraktif();
134. halamanSimulasi();
135. }
136. if (tombolAktif == "home"){
137. resetInteraktif();
138. halamanMenu();
139. }
140. }
141. canvas.onmousemove = null;
142. }
143. function keluarSimulasi(){
144. canvas.onmousedown = mouseDown;
145. canvas.onmouseup = mouseUp;
146. window.clearTimeout(mainTimer);
147. resetInteraktif();
148. halamanMenu();
149. }

```

3. Simpan dengan nama **simulasi-20.js**. Jangan lupa mengetikkan ekstensi *file* (.js) saat menyimpan, dan simpan pada *folder* tutorial-20.

Jalankan *file* **tutorial-20.html**, dengan cara membukanya di internet *browser*. Maka kode di atas akan menghasilkan aplikasi virtual lab dengan 3 jenis praktikum sebagai berikut :



Gambar 13.6 Hasil tutorial GUI fungsi halamanJudul



Gambar 13.7 Hasil tutorial GUI fungsi halamanNama



Gambar 13.8 Hasil tutorial GUI fungsi halamanMenu



Gambar 13.9 Hasil tutorial GUI fungsi halamanSimulasi

### e. Penjelasan Program

Dalam kode simulasi-20 di atas, masing-masing halaman pada dasarnya dilakukan penataan asset visual dengan menggunakan kode. Pada fungsi halamanJudul dilakukan penataan *background*, judul dan sebuah tombol, dimana tombol “mulai” jika ditekan akan membawa pengguna ke halaman selanjutnya (lihat baris 105-108), yaitu halamanNama.

Pada halamanNama, diletakkan sebuah `teksInput`, untuk membaca nama pengguna. Melalui pengaturan pada variabel `inputNama` (baris 19), aplikasi akan menjalankan fungsi `cekNamaPengguna` untuk mengetahui apakah nama sudah terinput dengan benar atau tidak, yang mana hasil dari pengecekan tersebut ditampilkan melalui variabel `popup1`. Apabila nama yang diinputkan sudah benar, maka pengguna akan dibawa ke halamanMenu.

Pada awal fungsi halamanMenu, digunakan kode `hapusLayar` dengan parameter tambahan :

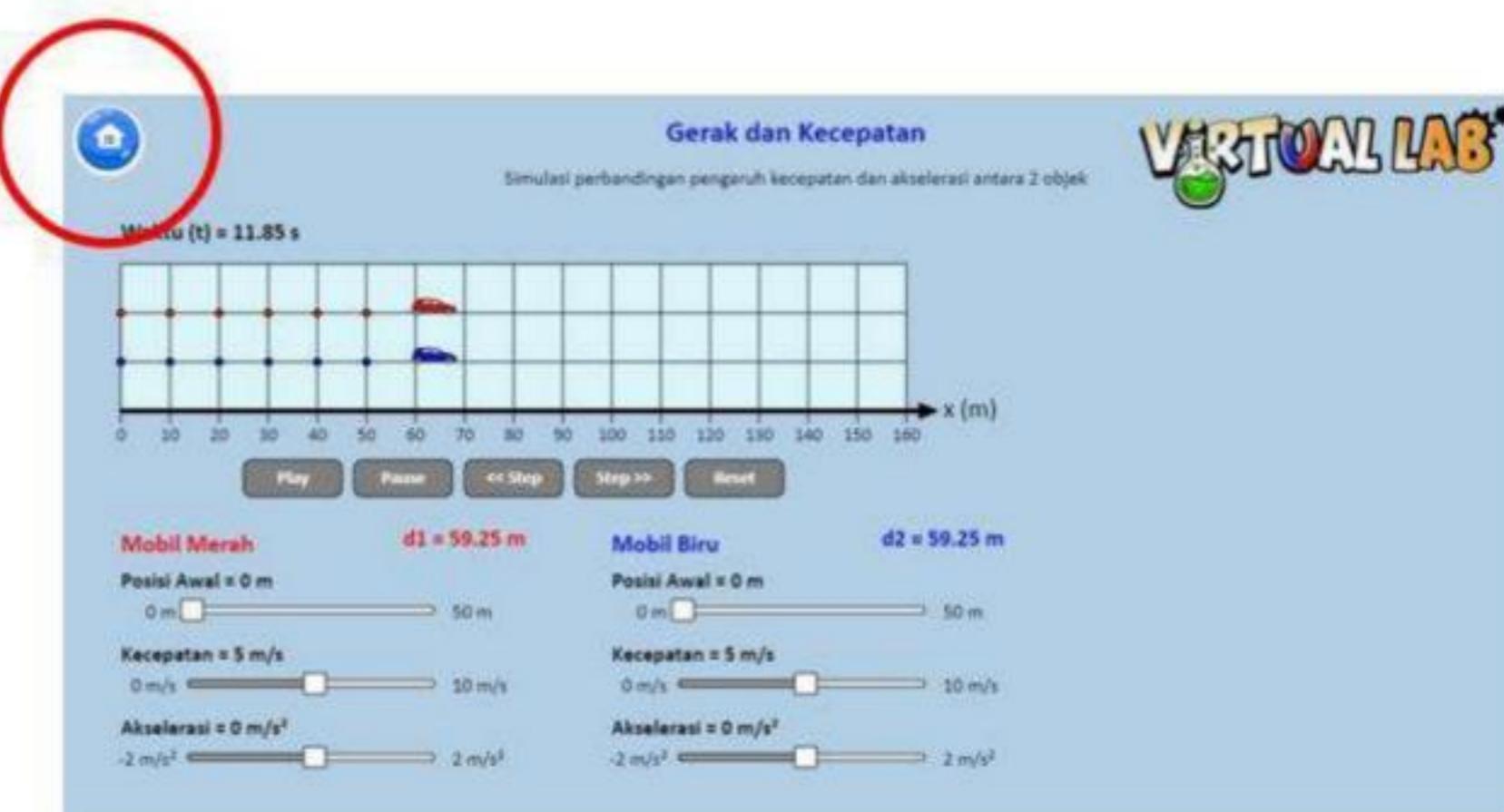
```
hapusLayar("#b3cfe5", {stat:"clear"});
```

Parameter `{stat:"clear"}` disini berfungsi untuk menghapus segala pengaturan yang dijalankan saat fungsi `hapusLayar` dijalankan. Perhatikanlah baris 76 pada fungsi `halamanSimulasi`, pada fungsi `hapusLayar` di baris 76 tersebut dilakukan pengaturan parameter `{stat:"run", func:tambahHome}`, yang berarti setiap kali fungsi `hapusLayar` dijalankan, maka fungsi `tambahHome` juga akan dijalankan. Fitur ini akan terus berjalan apabila tidak dihapus dengan perintah `{stat:"clear"}`. Ikuti penjelasan selanjutnya agar memahami lebih jelas maksudnya.

Pada fungsi `halamanMenu` terdapat 3 buah tombol, yang masing-masing akan membawa pengguna menuju ke `halamanSimulasi` (baris 121-135).

Selanjutnya, pada `halamanSimulasi` kode akan memanggil jenis simulasi yang dipilih. Jenis simulasi yang sudah disiapkan pada langkah a akan dieksekusi. Hal spesifik yang perlu diperhatikan pada fungsi `halamanSimulasi` adalah penambahan parameter `{stat:"run", func:tambahHome}`, yang menyebabkan dijalankannya fungsi `tambahHome` setiap kali fungsi `hapusLayar`

dipanggil, sehingga muncul tombol *home* di pojok kiri atas dan teks judul virtualLab di pojok kiri atas.



Gambar 13.10 Posisi tombol *home*

Pada fungsi `tambahHome`, dilakukan penambahan tombol :

```
tombolImg("home", 10, 10, 60, 60, dataGambar.tombol_home,
keluarSimulasi);
```

yang mana pada parameter terakhir, ditambahkan nilai `keluarSimulasi`, yang merupakan fungsi untuk berpindah dari halamanSimulasi ke halamanMenu (lihat penjelasan fungsi tombol pada Bab 6). Ketika pengguna menekan tombol "*home*" maka fungsi `keluarSimulasi` akan dijalankan dan aplikasi akan kembali ke halamanMenu. Pada tahapan ini perlu diperhatikan bahwa fungsi `hapusLayar` masih memiliki parameter tambahan yaitu menambahkan tombol *home*, sehingga tepat ketika fungsi `halamanMenu` dijalankan, maka parameter fungsi tersebut perlu dihapus dengan menambahkan parameter `{stat:"clear"}.`

Perhatikan teknik perpindahan halaman pada baris 105-135. Setiap kali akan berpindah halaman selalu dijalankan fungsi `resetInteraktif()`. Fungsi ini digunakan untuk menetralisir segala bentuk interaktivitas yang ada di halaman sebelumnya, sehingga interaktivitas pada halaman sebelumnya tidak akan berimbang pada halaman baru.

Fungsi keluarSimulasi akan dijalankan setiap kali tombol *home* ditekan. Hal yang perlu diperhatikan disini adalah, mengembalikan *listener mouse* dengan pengaturan ulang *onmousedown* dan *onmouseup*, serta menghapus kemungkinan *setTimeout* yang masih berjalan pada saat halamanSimulasi aktif.

### 13.3 Pengembangan Selanjutnya

Pada tahapan ini aplikasi laboratorium sederhana sudah dapat dikatakan siap digunakan meskipun masih memiliki fitur-fitur standar dan praktikum sederhana. Meskipun demikian diperlukan pengembangan lebih lanjut untuk mendapatkan aplikasi laboratorium virtual yang lengkap. Pengembangan fitur yang dapat dilakukan pada tahapan selanjutnya antara lain :

1. Penambahan jenis praktikum

Sebuah aplikasi laboratorium virtual yang baik memiliki beberapa jenis praktikum yang dapat dipilih oleh pengguna. Dalam kasus di buku ini, penambahan praktikum dapat mengacu pada langkah-langkah tutorial-20. Kode simulasi dapat dibuat secara terpisah, diuji terlebih dahulu, dan ketika simulasi berjalan dengan baik maka simulasi dapat digabungkan dengan file tutorial-20.

2. Penambahan halaman capaian pembelajaran / kompetensi yang diharapkan.

Secara umum aplikasi laboratorium virtual masuk ke dalam kategori media pembelajaran interaktif. Dalam hal ini diperlukan sebuah halaman yang mendeskripsikan tentang kompetensi yang diharapkan dari peserta didik setelah menggunakan aplikasi.

Metode menambahkan halaman capaian pembelajaran sama seperti menambahkan halaman pilihan menu (*fungsi halamanMenu*), hanya saja perlu ditambahkan teks (dengan fungsi *teksHTML*) dan sebuah tombol untuk kembali ke halaman sebelumnya.

3. Penambahan petunjuk praktikum (prosedur praktikum)

Petunjuk / prosedur praktikum merupakan salah satu elemen wajib dalam pelaksanaan praktikum laboratorium. Di dalam laboratorium virtual juga berlaku hal yang sama, diperlukan sebuah petunjuk teknis pelaksanaan simulasi tentang apa yang harus dilakukan oleh praktikan dalam menjalankan simulasi.

Menambahkan sebuah halaman petunjuk praktikum pada tutorial 20 dapat dilakukan dengan cara membuat fungsi baru yang akan dipanggil ketika pengguna memilih salah satu tombol simulasi (pada halamanMenu). Setelah pengguna menekan tombol pilihan simulasi, maka pengguna diarahkan menuju ke halaman petunjuk praktikum sebelum diarahkan ke halaman simulasi. Contoh dari penambahan halaman petunjuk / prosedur praktikum dapat anda temukan pada website [www.wandah.org/laboratorium-virtual](http://www.wandah.org/laboratorium-virtual)



Gambar 13.11 Contoh Penambahan Instruksi Praktikum

(sumber : laboratorium virtual biologi wandah.org)

#### 4. Penambahan sistem evaluasi praktikum

Untuk mengevaluasi praktikum dapat dilakukan dengan beberapa cara seperti penambahan halaman evaluasi (dengan sistem pertanyaan atau kuis). Menambahkan sistem evaluasi otomatis yang dilakukan dengan menambahkan variabel-variabel khusus untuk mencatat tindakan pengguna, ketepatan prosedur praktikum dan waktu praktikum, yang pada akhir simulasi dapat ditunjukkan hasil pencatatan tersebut. Atau melakukan evaluasi secara konvensional melalui pelaporan tertulis.



Gambar 13.12 Sistem review praktikum dalam laboratorium virtual

## BAB 14

### Penutup

#### 14.1 Langkah Selanjutnya

Pada tutorial-20, sebuah aplikasi laboratorium virtual secara sederhana telah dibuat dengan 3 jenis praktikum di dalamnya. Tutorial tersebut selanjutnya dapat anda publikasikan (*publishing*) kepada pengguna dengan beberapa alternatif pilihan sebagai berikut :

1. Publikasi melalui *website*

Proses publikasi yang paling sederhana yang dapat kita lakukan dengan tutorial/aplikasi yang telah kita buat adalah dengan mempublikasikannya dalam bentuk halaman *website*. Cara ini dilakukan dengan mengupload keseluruhan *file* ke *server*.

Untuk penggunaan website dengan *Content Management System* (CMS) seperti Wordpress atau Blogger, kita juga dapat mengupload seluruh *file* melalui *dashboard (admin)* dan membuat *post* baru dengan isi yang sama dengan *file* HTML di setiap tutorial.



Gambar 14.1 Mempublikasikan tutorial melalui *website*

2. Menggabungkan aplikasi dengan LMS

Pembelajaran *online* melalui website pada umumnya difasilitasi oleh *Learning Management System* (LMS). Dengan LMS kita dapat mengatur mata pelajaran atau perkuliahan dengan membuat kursus (*courses*) baru serta mengisi kursus baru tersebut dengan materi pelajaran. LMS populer seperti Edmodo, Moodle, Blackboard dan sejenisnya mendukung penambahan

materi pelajaran dalam format HTML. Melalui LMS tersebut kita dapat meng“embed” aplikasi laboratorium virtual kita, baik dengan cara mengupload keseluruhan konten (HTML, JS dan Gambar) ke dalam sebuah kursus atau menambahkannya secara terpisah dengan teknik *iframe*. Sebagai contoh, untuk LMS Moodle anda dapat membaca petunjuk penambahan konten pada link : [https://docs.moodle.org/22/en/Embedding\\_content](https://docs.moodle.org/22/en/Embedding_content)



Gambar 14.2 Penggabungan Lab-Virtual dengan LMS Moodle

### 3. Mempublish pada *platform mobile* (Android atau iOS)

Untuk menjadikan aplikasi yang kita buat menjadi sebuah aplikasi yang dapat dijalankan pada gawai (Android atau iOS), maka diperlukan beberapa penyesuaian dan *software* tambahan. Untuk *platform* Android misalnya, diperlukan beberapa installasi dan pengaturan *software* seperti Java Development Kit (JDK), Android SDK, Android Studio dan Apache Cordova. Proses publikasi ke format Android (APK), membutuhkan pengetahuan khusus, dan petunjuk operasional dapat diperoleh pada link berikut :

<https://cordova.apache.org/docs/en/5.0.0/guide/platforms/android/>

atau tutorial bahasa Indonesia : <https://www.petanikode.com/cordova/>



Gambar 14.3 Tutorial-20 pada Emulator Android

## 14.2 Meningkatkan Fitur Grafis 3 Dimensi

Pada tutorial di buku ini digunakan grafis 2D sederhana, sementara untuk menghasilkan imersi tinggi dan visualisasi yang mendekati laboratorium yang sesungguhnya, grafis pada aplikasi dapat ditingkatkan menjadi grafis 3 Dimensi. Terkait hal tersebut HTML canvas membutuhkan fitur webGL dan bantuan dari *library* yang mendukung fitur 3 dimensi. *Library* yang dapat kita gunakan untuk menghasilkan grafis 3 dimensi antara lain :

1. Babylon.js

*Library* Babylonjs memiliki fleksibilitas dalam pengolahan grafis 3 dimensi berbasis webGL, didukung dengan tutorial dan dokumentasi yang lengkap, serta mendukung luaran aplikasi grafis populer seperti Blender, 3Ds Max, Maya dan sebagainya. Babylon.js dapat diperoleh dari link berikut :

<https://www.babylonjs.com/>

2. Three.js

Seperti halnya Babylon, Threejs merupakan *library* yang menghususkan pada grafis 3 dimensi dengan fitur webGL. Dengan ukuran *library* yang kecil dan dokumentasi yang lengkap, Threejs dapat diperoleh dari link berikut :

<https://threejs.org/>

Terkait dengan grafis berjenis 3 dimensi, maka diperlukan aplikasi grafis 3D seperti Blender, 3Ds Max, Maya, Sketchup atau sejenisnya untuk menghasilkan objek 3 dimensi yang selanjutnya di "import" ke dalam aplikasi. Perhitungan simulasi pada dasarnya memiliki kesamaan dengan perhitungan pada mode 2 dimensi, namun untuk mendapatkan kesan kedalaman perhitungan dapat ditambahkan dengan elemen sumbu Z.



Gambar 14.4 Pengembangan grafis 3 dimensi

### **14.3 Saran terkait Pengembangan Laboratorium Virtual**

Pada penelitian terkait pelaksanaan praktikum sains, ditemukan bahwa siswa belum merasa siap atau cukup percaya diri untuk melaksanakan eksperimen di laboratorium. Masing-masing siswa memiliki pengetahuan dasar yang berbeda-beda terkait praktikum, ada yang telah mempersiapkan diri dengan membaca instruksi praktikum, dan ada yang membaca instruksi praktikum tepat sebelum praktikum dilakukan. Permasalahan lain adalah kurang atau tidak tersedianya fasilitas perangkat laboratorium, sehingga praktikum tidak dapat dilakukan dan hanya dideskripsikan melalui tulisan atau gambar. Permasalahan berikutnya dalam pelaksanaan praktikum laboratorium adalah keterbatasan waktu. Waktu yang telah ditentukan menutup peluang siswa untuk melakukan alternatif-alternatif tindakan, menutup peluang untuk mengulang apabila terjadi kesalahan, dan menutup peluang untuk mendalami proses praktikum untuk menghasilkan analisis yang mendalam.

Berbagai celah tersebut dapat ditutupi dengan penggunaan laboratorium virtual. Laboratorium virtual memang tidak akan dapat menggantikan laboratorium yang sesungguhnya, karena pengalaman langsung di lapangan (laboratorium) mutlak diperlukan dalam pembelajaran sains. Namun penggunaan laboratorium virtual dapat membantu mempersiapkan siswa sebelum praktikum (pra-praktikum), menutup kelemahan tidak tersedianya atau terbatasnya perangkat laboratorium, memungkinkan siswa untuk belajar dari kesalahan, memungkinkan pengulangan praktikum dengan waktu yang fleksibel, dan memungkinkan pendalaman materi eksperimen dengan lebih baik lagi.

Sebagai penutup, pengembangan laboratorium virtual dimungkinkan membawa nilai tambah bagi pembelajaran sains. Terlepas dari banyaknya aplikasi laboratorium virtual komersial maupun gratis yang telah dibuat oleh pengembang dari luar negeri, kita perlu mengembangkan laboratorium virtual yang sesuai dengan kurikulum pendidikan tanah air, melokalisasi konten, dan menggunakan sebagai media pendamping dalam pembelajaran sains. Akhirnya semoga buku sederhana yang memiliki banyak kekurangan ini membawa manfaat dan menjadi salah satu referensi dalam pengembangan laboratorium virtual.

## **Daftar Pustaka**

- [1] Hamalik, Oemar. 2008. *Kurikulum dan Pembelajaran*. Jakarta:Bumi Aksara.
- [2] Prince M. 2004. *Journal of Engineering Education* 93(3) 223
- [3] Yang K and Heh J. 2007. *Journal of Science Education and Technology* 16(5) 451
- [4] Daryanto. 2014. *Pendekatan Pembelajaran Saintifik Kurikulum 2013*. Yogyakarta: Penerbit Gava Media
- [5] Gega, Peter. 1982. *Science in Elementary Education*: 2d Ed. Wiley.
- [6] Arifin, Zainal. 2012. *Model Penelitian dan Pengembangan*, Bandung: PT Remaja Rosdakarya
- [7] Widowati dan Sutimin. 2007. *Buku Ajar Pemodelan Matematika*. Semarang:Universitas Diponegoro
- [8] Royce, Winston. 1970. *Managing the Development of Large Software Systems* (PDF), Proceedings of IEEE WESCON, 26 (August): 1–9



## Tentang Penulis



Wandah Wibawanto S.Sn., M.Ds. secara khusus mendalami pengembangan game dan aplikasi sejak tahun 2001. Menyelesaikan program S1 Desain Komunikasi Visual di Universitas Negeri Malang pada tahun 2006 dan S2 Magister Desain (*Game Tech/Media Digital*) ITB pada tahun 2012 dan saat buku ini ditulis sedang menempuh program S3 Pendidikan Seni Universitas Negeri Semarang.

Pada tahun 2006 bekerja di sebuah perusahaan game FreeOnline Games.com dan menghasilkan beberapa karya game yang cukup populer, di antaranya adalah Sim Taxi, Gangster Life, The Empire, dan beberapa game lainnya. Tahun 2008 memutuskan untuk mengembangkan game flash secara independen untuk beberapa game portal seperti Armorgames, Gameninja, Gamebooks, DailyFreeGames, dan Froyogames.com. Mendapatkan beberapa penghargaan di bidang aplikasi dan games seperti pemenang INAICTA 2009 kategori digital media, *Honorable mention Dictionary Games award*, *FOG daily game programing challenge* dan beberapa lainnya. Tahun 2014 bergabung menjadi dosen DKV Universitas Negeri Semarang, menjadi peneliti di bidang ekonomi kreatif, dan pengembangan media edukasi.

Beberapa buku yang telah ditulis terkait dengan pengembangan Aplikasi antara lain :

- Membuat Game dengan Flash (2003) Penerbit Andi, Yogya
- Dasar-dasar Pemrograman Flash Game (2006) e-book
- Membuat Flash Game 3D (2013) Penerbit Andi, Yogya
- Desain dan Pemrograman Multimedia Interaktif (2017) Penerbit Cerdas Ulet Kreatif, Jember
- Membuat bermacam-macam Game *Android* dengan Adobe Animate (2019) Penerbit Andi, Yogya
- Game Edukatif RPG (2020), Penerbit LPPM UNNES, Semarang

Beberapa buku terkait ekonomi kreatif antara lain :

- *Book chapter "Kolase Pemikiran Ekonomi Kreatif"* (2017)
- Pendampingan OPD dalam Pengembangan Ekonomi Kreatif Kab/Kota (2018)
- Integrasi Rencana Induk Pengembangan Ekonomi Kreatif (Rindekraf) dalam RPJMD Kab/Kota (2019)

Beberapa karya aplikasi, tutorial dan materi perkuliahan dapat ditemukan di situs [wandah.com](http://wandah.com), [wandah.org](http://wandah.org) dan [froyogames.com](http://froyogames.com). Penulis dapat dihubungi via email di [wandah@wandah.com](mailto:wandah@wandah.com) atau [wandah@mail.unnes.ac.id](mailto:wandah@mail.unnes.ac.id)

# Wandah Wibawanto

# LABORATORIUM VIRTUAL

## KONSEP DAN PENGEMBANGAN

## SIMULASI FISIKA

Istilah laboratorium virtual merupakan istilah yang menjadi *hype* dalam 1 dekade terakhir. Keberadaan teknologi seperti *augmented reality* dan *virtual reality* menjadikan banyak pengembang aplikasi merilis laboratorium virtual dalam berbagai format, mulai dari *platform* umum seperti *mobilephone* dan website sampai dengan aplikasi yang membutuhkan perangkat khusus seperti *Head Mounted Display* untuk mensimulasikan laboratorium virtual dalam bentuk realitas maya.

Sementara itu, pengembangan laboratorium virtual oleh pengembang lokal atau pengajar tanah air, masih relatif minim. Hal ini mungkin disebabkan karena minimnya referensi, atau sudah merasa nyaman dengan penggunaan aplikasi yang telah dibuat oleh pengembang luar negeri. Oleh karena itu buku ini dimaksudkan untuk menjadi salah satu referensi tentang konsep dan pengembangan laboratorium virtual dengan tujuan untuk menumbuhkan ketertarikan kepada siapapun dalam mengembangkan media pembelajaran sains berupa laboratorium virtual, yang mungkin dapat digunakan dalam ruang lingkup kecil yaitu pembelajaran di kelas atau bahkan dikembangkan lebih lanjut untuk menjadi aplikasi yang digunakan secara global.

Simulasi dalam buku ini dibuat menggunakan bahasa pemrograman Javascript dan ditampilkan melalui HTML 5 Canvas, sehingga mudah untuk digunakan dan tidak membutuhkan perangkat lunak berbayar. Penjelasan di dalamnya cukup sederhana dan diharapkan dapat dipahami oleh pembelajar pemula. Luaran buku ini berupa beberapa simulasi fisika dapat dikembangkan lebih lanjut atau dapat diadaptasi untuk dikembangkan dalam bahasa pemrograman lain. Kritik dan saran guna perbaikan sangat diharapkan. Akhirnya semoga buku ini membawa manfaat bagi pembaca serta bagi pengembangan virtual laboratorium di Indonesia.

### PENERBIT LPPM UNNES

Gedung Prof. Retno Sriningsih Satmoko Lantai 1  
Kampus Sekaran Gunungpati Semarang 50229

Telp/Fax : (024) 8508089  
Email : lppm@mail.unnes.ac.id

ISBN 978-623-7618-84-3



9 78623 618843