UNIVERSITAS SUNAN GRESIK
KOKOH MELESAT MENDUNIA

Komplek PPM Sunanul Muhtadin, Kertosono
Sidayu Gresik Jawa Timur Indonesia
universitassunangresik@gmail.com
0812-343-222

Simulasi 2D Gerak dan Tumbukan Bola dengan Pendekatan Fisika

Sofyan Gio Verdiansyah
Fakultas Teknologi dan Rekayasa
Teknik Robotika dan Kecerdasan Buatan
Universitas Sunan Gresik
Gresik, Indonesia
gioverdiansyh@gmail.com

***Abstrak*— Penelitian ini bertujuan untuk mengembangkan simulasi interaktif dua dimensi yang memodelkan perilaku fisika pada sebuah bola sebagai objek utama dengan prinsip dasar fisika, khususnya Hukum Newton II dan Gerak Lurus Berubah Beraturan (GLBB). Proyek ini dibuat sebagai bagian dari tugas mata kuliah Fisika Dasar dan dimaksudkan untuk membantu mahasiswa memahami penerapan konsep fisika dalam konteks digital melalui pemrograman. Dalam simulasi ini, gerak objek dianalisis menggunakan persamaan dasar Hukum Newton II serta konsep GLBB untuk mengatur percepatan dan kecepatan pada bola. Selain itu, permainan dilengkapi dengan algoritma bot sederhana yang merespons posisi bola secara dinamis. Metode pengembangan dilakukan menggunakan teknologi JavaScript (JS) dan HyperText Markup Language 5 (HTML5) Canvas sebagai media visualisasi. Hasil implementasi menunjukkan bahwa simulasi berjalan sesuai dengan hukum fisika yang telah diterapkan, meski memerlukan konversi nilai agar konsep fisika didunia nyata berjalan didunia digital. Penelitian ini diharapkan dapat memberikan dampak positif bagi pelajar dalam pemahaman fisika dasar dan sekaligus dapat melatih keterampilan logika pemrograman.* (*Abstract*)

Keywords—collision, GLBB, Newton's law, simulation, physics-based game.

I. PENDAHULUAN

Pada mata kuliah Fisika Dasar, mahasiswa sering mengalami kesulitan dalam memvisualisasikan penerapan konsep Hukum Newton II dan GLBB. Oleh karena itu, proyek ini dikembangkan sebagai bagian dari tugas akhir setengah semester dengan tujuan memberikan pemahaman berbasis simulasi digital melalui pemrograman.

Penelitian ini secara khusus berfokus pada perilaku bola sebagai objek utama simulasi. Konsep fisika seperti percepatan (a), gaya (F), dan massa (m) dimodelkan menggunakan persamaan dasar $F = m \cdot a$ untuk mengatur perubahan kecepatan bola, sedangkan GLBB digunakan untuk memodelkan perubahan kecepatan dalam kurun waktu tertentu. Untuk memperkuat realisme simulasi, komponen fisika lain seperti gaya gesek udara, gaya gesek bidang datar, dan pantulan dengan koefisien restitusi juga diimplementasikan.

Simulasi ini dikembangkan menggunakan bahasa pemrograman JS dan HTML5 Canvas, yang berfungsi sebagai media visualisasi dua dimensi. Selain pergerakan bola, proyek ini juga mencakup implementasi bot yang merespons posisi bola dalam area yang telah ditentukan. Pendekatan ini tidak hanya menunjukkan penerapan konsep dasar fisika, tetapi juga memperlihatkan bagaimana logika dan algoritma dapat diintegrasikan ke dalam sistem gerak berbasis fisika

Beberapa penelitian sebelumnya menunjukkan bahwa penggunaan simulasi berbasis game dapat meningkatkan pemahaman konsep fisika karena sifatnya yang interaktif dan memungkinkan eksplorasi eksperimen secara langsung tanpa alat laboratorium fisik. Dengan mengadaptasi model serupa, penelitian ini bertujuan untuk memberikan alternatif media pembelajaran yang lebih intuitif, terutama bagi mahasiswa yang memiliki latar belakang pemrograman.

Kontribusi utama penelitian ini adalah:

1. Menyajikan model simulasi fisika dua dimensi yang memfokuskan pada dinamika bola sesuai hukum Newton dan GLBB.
2. Mengintegrasikan perhitungan fisika nyata (collision, gaya, percepatan, gesekan, restitusi) ke dalam implementasi kode secara langsung.
3. Membuktikan bahwa pendekatan interaktif berbasis pemrograman dapat meningkatkan pemahaman konsep fisika secara praktis dan konseptual.

Struktur artikel ini terdiri atas: bagian metode yang menjelaskan model fisika dan pendekatan pemrograman, bagian hasil dan pembahasan yang menyajikan dampak implementasi simulasi terhadap dinamika bola, dan bagian kesimpulan yang merangkum hasil penelitian dan potensi pengembangannya.

II. METODE

Metode penelitian ini bersifat **kualitatif–kuantitatif terapan** yang menggabungkan pengembangan perangkat lunak (software engineering) dan simulasi fisika numerik untuk memodelkan dinamika bola dalam arena 2 dimensi (2D). Proses penelitian meliputi tahap perancangan alur simulasi, implementasi perangkat lunak, kalibrasi konversi nilai digital ke rumus, pengujian skenario, dan evaluasi kesesuaian hasil terhadap ekspektasi teori.

a) Desain Penelitian

Penelitian ini menggunakan pendekatan **eksperimen simulatif**: membuat model numerik yang merepresentasikan fenomena fisika (tumbukan, gaya, gesekan, restitusi),

menjalankan simulasi pada berbagai kondisi kontrol, dan menganalisis keluaran (trajectory, kecepatan, respons tumbukan) untuk menilai kesesuaian dengan teori dasar Hukum Newton II dan GLBB. Projek ini merupakan modifikasi dari projek latihan sebelumnya, dengan penambahan komponen fisika dan kalibrasi numerik. Untuk membantu proses konversi nilai fisika ke domain digital, penulis menggunakan alat bantu AI sebagai sumber saran teknis.

b) Alat dan Lingkungan Pengembangan

Penelitian ini tidak menggunakan perangkat keras fisik, tetapi seluruh proses simulasi dilakukan dalam lingkungan pemrograman berbasis web. Adapun teknologi dan software yang digunakan pada tabel di bawah ini.

Tabel 1. Teknologi yang digunakan

No	Teknologi	Fungsi
1.	**HTML5 Canvas**	sebagai media untuk menampilkan visualisasi 2D.
2.	**JavaScript**	sebagai bahasa pemrograman utama untuk logika simulasi.
3.	**CSS**	untuk pengaturan layout tampilan sederhana.
4.	Visual Studio Code	sebagai IDE atau Editor penulisan kode-kode.
5.	Web browser (Chrome/Firefox)	sebagai tempat visualisasi dan simulasi di tampilkan.

Struktur folder terdiri dari berkas *index.html*, *style.css*, dan *script.js* yang berperan dalam membangun antarmuka dan logika permainan.

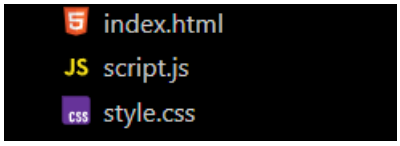


Fig. 1. Struktur Folder

c) Pemodelan Fisika

Model fisika difokuskan pada bola sebagai objek utama dan meliputi:

1. **Deteksi Tumbukan (Circle Collision):** Menghitung jarak antar pusat lingkaran; tumbukan terjadi bila jarak ≤ sum radius [1].
2. **Normalisasi Vektor:** Menggunakan vektor unit untuk menentukan arah gaya agar jarak tidak memengaruhi magnitudo gaya.
3. **Hukum Newton II:** Percepatan bola dihitung dari $a = \frac{F}{m}$, dimana F meliputi gaya dorong pemain (Fpush), gaya gesek papan, dan gaya drag udara.
4. **GLBB & Integrasi Numerik:** Perubahan kecepatan dan posisi dihitung dengan metode Euler diskrit per frame.
5. **Gesekan:** Model gesek papan disederhanakan sebagai percepatan melawan arah gerak, sedangkan drag udara dimodelkan sebagai proporsional terhadap kecepatan atau konstanta linier (disesuaikan selama kalibrasi).
6. **Pantulan (Restitusi):** Komponen kecepatan tegak lurus dinding dikalikan koefisien restitusi e untuk mensimulasikan kehilangan energi saat tumbukan.

d) Alur dan Pengembangan

Pada alur simulasi ini dapat digambarkan dengan flowchart dibawah:

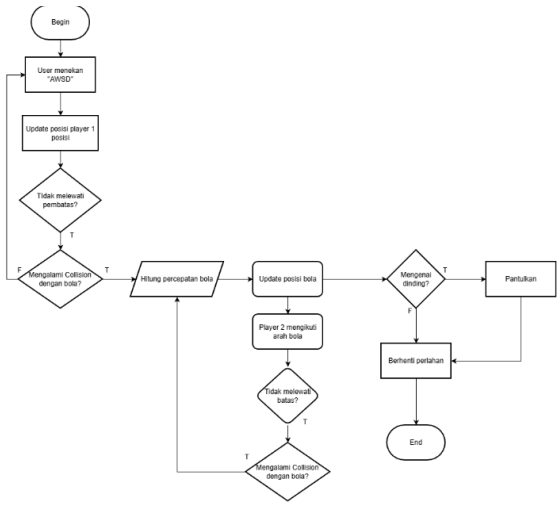


Fig. 2. Flowchart simulasi berjalan

Pada tahap-tahapan implementasi di jabarkan sebagai berikut:

1. **Penulisan awal kode** HTML dan CSS sebagai pondasi tampilkan canvas.

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>TA Fisika Semester 1</title>
  <link rel="stylesheet" href="./css/style.css">
</head>

<body>
  <canvas id="main"></canvas>

  <script src="./js/script.js"></script>
</body>

</html>
```

Fig. 3. Isi file HTML

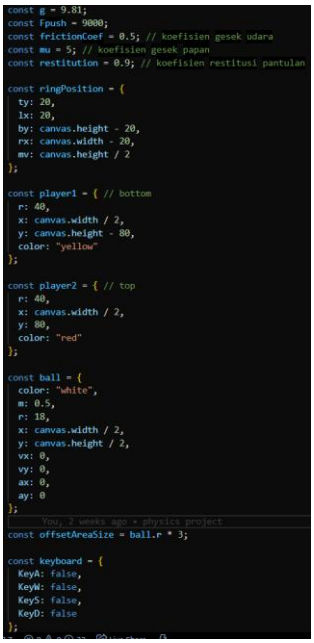
Kode diatas adalah penghubung file css dan JS serta tag canvas sebagai antarmuka simulasi.

```
body {
  padding: 0;
  margin: 0;
  overflow: hidden;
  display: flex;
  justify-content: center;
}
```

Fig. 4. Isi file CSS

Pada kode CSS hanya bertugas menengahkan tag canvas di HTML serta menghilangkan padding, margin dan menyembunyikan elemen yang overflow.

2. **Perancangan struktur kode** (inisialisasi nilai konstan simulasi dan fisika serta pemisahan fungsi draw, update, input dan physics).



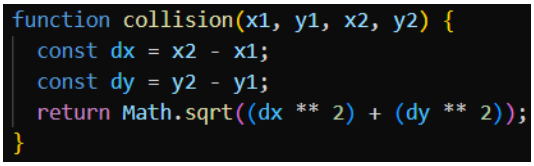
```
const g = 9.81;
const Fpush = 9000;
const frictionCoef = 0.5; // koefisien gesek udara
const mu = 5; // koefisien gesek papan
const restitution = 0.9; // koefisien restitusi pantulan

const ringPosition = {
  ty: 20,
  lx: 20,
  by: canvas.height - 20,
  rx: canvas.width - 20,
  mv: canvas.height / 2
};

const player1 = { // bottom
  r: 40,
  x: canvas.width / 2,
  y: canvas.height - 80,
  color: "yellow"
};

const player2 = { // top
  r: 40,
  x: canvas.width / 2,
  y: 80,
  color: "red"
};

const ball = {
  color: "white",
  m: 0.5,
  r: 18,
  x: canvas.width / 2,
  y: canvas.height / 2,
  vx: 0,
  vy: 0,
  ax: 0,
  ay: 0
};
      You, 2 weeks ago • physics project
const offsetAreaSize = ball.r * 3;

const keyboard = {
  KeyA: false,
  KeyW: false,
  KeyS: false,
  KeyD: false
};
```

Fig. 5. Inisialisasi variabel 1

```
function update(dt) {
  playerController(dt);
  botMoveLogic(dt);
  handleCollisions(dt);
  updateBall(dt);
}

function draw() {
  drawBackground();
  drawRing();
  drawOffsetArea();
  drawPlayersBall();
}

// loop utama
let lastTime = performance.now();

function main(now) {
  const dt = (now - lastTime) / 1000;
  lastTime = now;

  update(dt);
  draw();

  // renderFormulas();
  requestAnimationFrame(main); // default 60FPS
}

requestAnimationFrame(main);
```

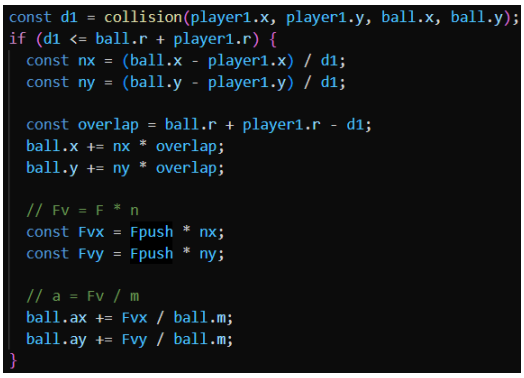
Fig. 6. Fungsi-fungsi bagaimana game dimulai

3. **Implementasi fungsi fisika** (collision, normalisasi, hit response, friction, drag, restitution) kedalam program.



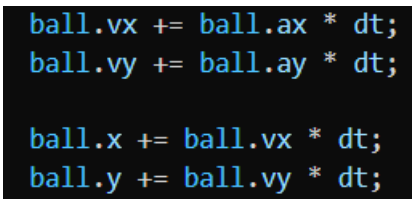
```
function collision(x1, y1, x2, y2) {
  const dx = x2 - x1;
  const dy = y2 - y1;
  return Math.sqrt((dx ** 2) + (dy ** 2));
}
```

Fig. 7. Fungsi collision



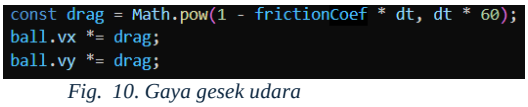
```
const d1 = collision(player1.x, player1.y, ball.x, ball.y);
if (d1 <= ball.r + player1.r) {
  const nx = (ball.x - player1.x) / d1;
  const ny = (ball.y - player1.y) / d1;

  const overlap = ball.r + player1.r - d1;
  ball.x += nx * overlap;
  ball.y += ny * overlap;

  // Fv = F * n
  const Fvx = Fpush * nx;
  const Fvy = Fpush * ny;

  // a = Fv / m
  ball.ax += Fvx / ball.m;
  ball.ay += Fvy / ball.m;
}
```

Fig. 8. Logika ketika collision: normalisasi, koreksi overlap, hitung percepatan



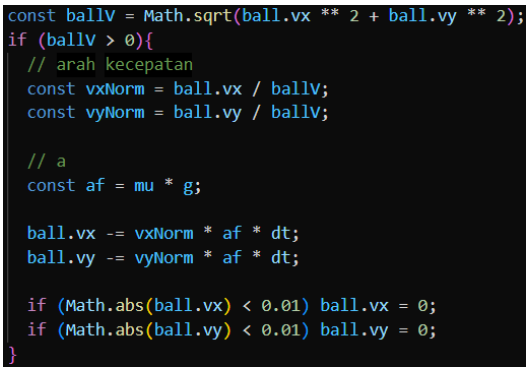
```
ball.vx += ball.ax * dt;
ball.vy += ball.ay * dt;

ball.x += ball.vx * dt;
ball.y += ball.vy * dt;
```

Fig. 9. Update percepatan dan posisi



```
const drag = Math.pow(1 - frictionCoef * dt, dt * 60);
ball.vx *= drag;
ball.vy *= drag;
```

Fig. 10. Gaya gesek udara



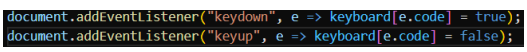
```
const ballV = Math.sqrt(ball.vx ** 2 + ball.vy ** 2);
if (ballV > 0){
  // arah kecepatan
  const vxNorm = ball.vx / ballV;
  const vyNorm = ball.vy / ballV;

  // a
  const af = mu * g;

  ball.vx -= vxNorm * af * dt;
  ball.vy -= vyNorm * af * dt;

  if (Math.abs(ball.vx) < 0.01) ball.vx = 0;
  if (Math.abs(ball.vy) < 0.01) ball.vy = 0;
}
```

Fig. 11. Gaya gesek papan mendatar

```
// wall bounce
if (ball.x - ball.r < ringPosition.lx) {
  ball.x = ringPosition.lx + ball.r; // bola agar langsung menempel dan pantulan jadi lebih akurat / elastis
  ball.vx *= -restitution;
}
if (ball.x + ball.r > ringPosition.rx) {
  ball.x = ringPosition.rx - ball.r;
  ball.vx *= -restitution;
}
if (ball.y - ball.r < ringPosition.ty) {
  ball.y = ringPosition.ty + ball.r;
  ball.vy *= -restitution;
}
if (ball.y + ball.r > ringPosition.by) {
  ball.y = ringPosition.by - ball.r;
  ball.vy *= -restitution;
}
```

Fig. 12. Pantulan pada dinding

4. **Penanganan input pemain** (key events):



```
document.addEventListener("keydown", e => keyboard[e.code] = true);
document.addEventListener("keyup", e => keyboard[e.code] = false);
```

Fig. 13. Fungsi pemicu gerak player

```
function playerController(dt) {
  const moveSpeed = 350; // px/s

  if (keyboard.KeyW && player1.y > (ringPosition.my + player1.r))
    player1.y -= moveSpeed * dt;

  if (keyboard.KeyS && player1.y < (ringPosition.by - player1.r))
    player1.y += moveSpeed * dt;

  if (keyboard.KeyA && player1.x > (ringPosition.lx + player1.r))
    player1.x -= moveSpeed * dt;

  if (keyboard.KeyD && player1.x < (ringPosition.rx - player1.r))
    player1.x += moveSpeed * dt;
}
```

Fig. 14. Logika penggerak playe

Dimana hasil dari *e.code* adalah kode keyboard pada JavaScript, dimana jika kode sama dengan variabel keyboard maka akan mengganti nilai menjadi true atau false.

```
function botMovelogic(dt) {
  const speed = 350; // px/s

  if (ball.y < ringPosition.my - 20) {
    if (ball.x < player2.x) player2.x -= speed * dt;
    if (ball.x > player2.x) player2.x += speed * dt;
    if (ball.y < player2.y) player2.y -= speed * dt;
    if (ball.y > player2.y) player2.y += speed * dt;
  } else if (ball.y > ringPosition.my + 20) {
    if (ball.x < player2.x && player2.x > (ringPosition.lx + player2.r))
      player2.x -= (speed * dt) / 2;

    if (ball.x > player2.x && player2.x < (ringPosition.rx - player2.r))
      player2.x += (speed * dt) / 2;

    if (ball.y < (ringPosition.my + (ringPosition.my / 2.5)) - 20 && (player2.y < (ringPosition.my - (player2.r + 5))))
      player2.y += (speed * dt);

    if (ball.y > player2.y && player2.y > (ringPosition.ty + (player2.r + 20)))
      player2.y -= (speed * dt) / 5;
  }
}
```

Fig. 15. Algoritma bot

Sedangkan untuk logika bot dimana menggunakan algoritma yang ditetapkan untuk mengikuti bola. Contoh jika bola ke-kiri maka bot juga akan ke-kiri.

5. **Loop utama**: *main()* memanggil *update()* (fisika & logika) lalu *draw()* (rendering) secara berulang sesuai frame rate bawaan browser (60 FPS). Ditunjukkan pada Fig. 3.
6. **Penambahan mekanisme stabilitas**: ring batas arena untuk mencegah akumulasi gaya berlebih dan koreksi overlap.

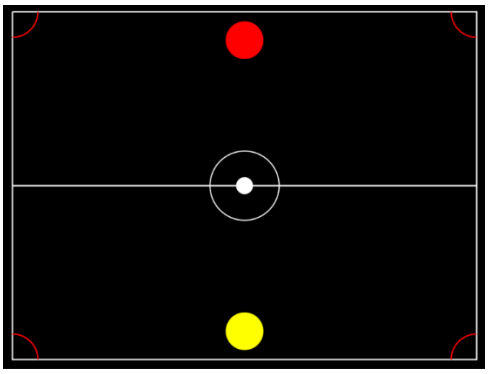


Fig. 16. Ring simulasi

```
// Offset Area Collision
if (collision(ball.x, ball.y, 20, 20) <= (ball.r + (offsetAreaSize))) reset();
if (collision(ball.x, ball.y, canvas.width - 20, 20) <= (ball.r + (offsetAreaSize))) reset();
if (collision(ball.x, ball.y, canvas.width - 20, canvas.height - 20) <= (ball.r + (offsetAreaSize))) reset();
if (collision(ball.x, ball.y, 20, canvas.height - 20) <= (ball.r + (offsetAreaSize))) reset();
```

Fig. 17. Logika area offset

e) Kalibrasi dan Konversi Nilai Fisik ke Digital

Karena ranah simulasi menggunakan piksel dan frame (bukan unit SI m, kg, s secara langsung), maka perlu dilakukan langkah kalibrasi:

1. **Mapping skala:** Menetapkan hubungan antara piksel dan meter pada canvas.
2. **Penentuan dt nyata:** Menggunakan dt (detik per frame) yang diukur dari fungsi JavaScript *performance.now()* untuk menjaga konsistensi waktu nyata.
3. **Penentuan konstanta skala:** Mengalibrasi konstanta seperti Fpush, koefisien gesek, dan drag agar respons bola terlihat natural. Contoh: Fpush awal diuji hingga 9000 Newton (N) pada setting digital (nilai besar karena skala piksel/time), kemudian diturunkan atau dikompensasikan melalui skala massa/konversi.
4. **Validasi kalibrasi:** Melakukan eksperimen iteratif (trial-and-error) untuk menyesuaikan parameter hingga perilaku bola menyerupai ekspektasi teori (mis. laju pelambatan, jarak pantulan).

f) Skenario Uji dan Pengumpulan Data

Pengujian dilakukan dengan beberapa skenario untuk mengamati perilaku bola:

- **Skenario A:** Dorongan tunggal dari player ke bola pada posisi diam lalu ukur percepatan awal, kecepatan puncak, jarak tempuh.
- **Skenario B:** Tumbukan bola dengan dinding dengan variasi koefisien restitusi (e) dan ukur kecepatan sebelum & sesudah pantulan.
- **Skenario C:** Pergerakan bola pada permukaan dengan μ (koefisien gesek) dan konstanta drag lalu amati laju pelambatan.
- **Skenario D:** Interaksi dengan bot (rule-based), observasi respons bot dan stabilitas permainan. Setiap skenario dijalankan beberapa kali untuk memastikan konsistensi; keluaran yang dicatat meliputi posisi (x,y), kecepatan (vx, vy), percepatan (ax, ay), waktu kejadian tumbukan, dan kondisi parameter.

g) Validasi dan keterbatasan metode

- **Validitas internal**: Model memanfaatkan persamaan fisika dasar, namun hasil bergantung pada kalibrasi numerik valid untuk demonstrasi pendidikan, bukan sebagai pengganti eksperimen fisik nyata.
- **Keterbatasan numerik**: Penggunaan metode Euler sederhana memperkenalkan error numerik; nilai dt harus cukup kecil atau metode integrator yang lebih baik diperlukan untuk akurasi tinggi.
- **Asumsi simplifikasi**: Drag dan gesek disederhanakan; faktor seperti rotasi bola, deformasi material, atau turbulensi udara tidak dimodelkan.
- **Sumber data & referensi**: Projek ini merupakan adaptasi dari projek latihan sebelumnya dan modifikasi oleh penulis; tidak menggunakan referensi literatur formal, tetapi memanfaatkan saran teknis dari AI untuk strategi konversi nilai digital.

A. *Penulis dan Afiliasi*

Sofyan Gio Verdiansyah*

Fakultas Teknologi dan Rekayasa, Program Studi Teknik Robotika dan Kecerdasan Buatan, Universitas Sunan Gresik

*Corresponding author: gioverdiansyh@gmail.com

III. *HASIL DAN PEMBAHASAN*

A. *Unit*

Seluruh besaran fisika dalam simulasi menggunakan satuan SI (MKS), termasuk massa (kg), gaya (N), percepatan (m/s²), dan kecepatan (m/s). Representasi posisi pada canvas menggunakan satuan piksel (px) dan hanya digunakan sebagai skala visual.

B. *Persamaan dan Rumus*

Dalam simulasi ini menggunakan beberapa konsep dan persamaannya di antara lainnya adalah: Collision, Normalisasi, Overlap, Kinematika GLBB, Hukum Newton II, Gesekan dan Pantulan.

1. Collision

Collision digunakan untuk mengukur apakah player dan bola akan saling bertumbukan atau bertabrakan. [1]

Rumusnya:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Keterangan:

d : Jarak antara kedua titik pusat.

$(x_1,\ y_1)$: posisi lingkaran pertama

$(x_2,\ y_2)$: posisi lingkaran kedua

Jika nilai d:

$$d \leq\ r_1 + r_2$$

r_1: Jari-jari lingkaran pertama

r_2: Jari-jari lingkaran kedua

Penerapan pada kode ditampilkan pada gambar Fig. 7.

Ketika player dan bola mengalami tumbukan, maka kita bisa melakukan perhitungan seperti GLBB atau Hukum Newton II.

2. Normalisasi

Dikarenakan nilai digital tidak di masukkan kedalam rumus Hukum Newton II secara langsung maka diperlukan normalisasi atau konversi, jika tidak, maka saat player berpindah gaya dapat berubah karena di pengaruhi oleh jarak.

Arah vector player dan bola:

$$\vec{r} = (x_2 - x_1, y_2 - y_1)$$

$\vec{r}$ adalah vector arah dari koordinat player x dan y.

Normalisasi artinya membuat vektor memiliki panjang 1 (satuan). Misal panjang vector $\vec{r}$ adalah:

$$|\vec{r}| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Maka **unit normal** (vektor arah dengan panjang 1) adalah [3], [4], [5]:

$$\hat{n} = \frac{\vec{r}}{|\vec{r}|} = \left(\frac{x_2 - x_1}{|\vec{r}|}, \frac{y_2 - y_1}{|\vec{r}|}\right)$$

Arti:

$|\vec{r}|$: panjang vector arah

$\hat{n}$: hasil unit normal

Penerapan pada kode ditampilkan pada gambar Fig. 8 paling atas.

3. Overlap

Jika kedua lingkaran yakni player dan bola saling tumpang tindih atau overlap maka kita bisa mencegahnya dengan cara: [1], [6]

$$o_{AB} = r_A + r_B - d$$
$$\vec{p}_B \leftarrow \vec{p}_B + (\hat{n} \cdot o_{AB})$$

Arti:

o_{AB} : Hasil nilai perhitungan Overlap antar A dan B

r : adalah radius baik A dan B

d : Jarak titik temu / collision

$\vec{p}_B$: Posisi baik X atau Y nantinya

Penerapan pada kode ditampilkan pada gambar Fig. 8 ditengah.

4. Kecepatan

Pada kecepatan saya menggunakan rumus GLBB untuk mencari kecepatan akhirnya, dimana rumusnya adalah [7]:

$$v = v_0 + a \cdot t$$

Arti:

v: Kecepatan

v_0: Kecepatan awal

a: Percepatan

t: Waktu

Penerapan pada kode ditampilkan pada gambar Fig. 9.

5. Hukum Newton II

Pada Hukum Newton II saya menggunakannya hanya untuk mencari percepatan berdasarkan gaya dorong dan massa. [7]

Rumus awalnya:

$$F = m\ \cdot a$$

Rumus percepatannya:

$$a = \frac{F}{m}$$

Arti:

a : Percepatan

F : Gaya dorong

m : Massa bola

Penerapan pada kode ditampilkan pada gambar Fig. 8.

6. Gesekan

Pada game saya menambahkan gaya gesek udara dan papan mendatar dimana agar membuat bola berhenti dengan lebih halus.[2]

1. Gaya gesek papan

Dimana rumus awalnya adalah:

$$F_g = \mu \cdot N$$

Keterangan:

- F_g : gaya gesek (Newton, N)
- μ : koefisien gesek antara papan dan benda (tanpa satuan)
- N = gaya normal (Newton, N)

Karena bola bergerak di papan mendatar dimana

$$N = m \cdot g$$

Sehingga

$$F_g = \mu \cdot m \cdot g$$

Dan karena gaya gesek akan bekerja saat bola bergerak maka
μ (koefesiensi gaya gesek) menjadi μ_k dimana μ_k adalah koefisien gesek kinetik.

Dan karena pergerakan bola adalah vector maka kita perlu menjadikannya besaran vector:

$$\overrightarrow{F_g} = -\mu_k \cdot m \cdot g \cdot \hat{v}$$

dengan $\hat{v}$ adalah **unit vector arah kecepatan** bola.
Secara vector:

$$\vec{v} = (v_x, v_y)$$

Jika di hitung besaran kecepatannya adalah:

$$|\vec{v}| = \sqrt{v_x^2 + v_y^2}$$

Maksud dari rumus di atas adalah agar tahu seberapa cepat bola bergerak (tanpa peduli arah), dan untuk membuat **vektor unit arah gerak** $\hat{v}$.
Dengan

$$\hat{v} = \frac{\vec{v}}{|\vec{v}|} = \left(\frac{v_x}{|\vec{v}|}, \frac{v_y}{|\vec{v}|}\right)$$

Karena $\vec{F} = m\vec{a} \Rightarrow \overrightarrow{a_g} = \frac{\overrightarrow{F_g}}{m} = \frac{\mu_k mg}{m}$ maka:
Massa di atas dan di bawah bisa saling di hilangkan menjadi:

$$\overrightarrow{a_g} = -\mu_k g\hat{v}$$

Ini berarti percepatan atau perlambatan akibat gesekan **arahnya berlawanan dengan arah gerak**, dan **besarannya tetap sebesar** $\mu_k g$

Pada kode dapat ditulis:

$$\vec{v}_{t+\Delta t} = \vec{v}_t - (\mu_k g\hat{v})\Delta t$$

Keterangan:
$\vec{v}_{t+\Delta t}$: kecepatan bola pada waktu berikutnya.
$\vec{v}_t$: kecepatan dalam bentuk vector bola saat ini.
Δt : selang waktu per frame.
Pada tanda minus (-) menunjukkan gesekan akan memperlambat gerak bola.

yang merupakan **integrasi numerik (metode Euler)** dari rumus percepatan:

$$\frac{d\vec{v}}{dt} = \mu_k g\hat{v}$$

Pada $\frac{d\vec{v}}{dt}$ merupakan turunan arah kecepatan vector terhadap waktu atau per-frame.

Penerapannya dapati dilihat pada gambar Fig. 11.

2. Gaya gesek udara

Pada benda yang bergerak dalam udara atau air, ada **gaya hambat (drag force)** yang arahnya **berlawanan dengan arah gerak**.
Model drag untuk kecepatan tinggi biasanya:

$$F_d = -kv^2$$

Tanda minus (−) artinya gaya drag melawan arah kecepatan.

- F_d : gaya hambat (N)
- k : konstanta hambatan (N·s/m)
- v : kecepatan (m/s)

Jadi Gaya gesek sebanding dengan kecepatan.

Dengan $k = \frac{1}{2} C_d \rho A v^2$

Keterangan:

- C_d : koefisien drag (tanpa satuan)
- ρ : massa jenis fluida (kg/m³)
- A : luas penampang (m²)
- v : kecepatan (m/s)

Berdasakan rumus Hukum Newton II

$$\sum F = m \cdot a$$

Karena percepatan (a) pada game adalah turunan (d) kecepatan terhadap waktu $a = \frac{dv}{dt}$ maka:

$$\sum \boldsymbol{F = m\frac{dv}{dt}}$$

Jadi dapat di tulis juga:

$$\boldsymbol{m\frac{dv}{dt} = -kv^2}$$

Dalam banyak simulasi (terutama game engine atau grafika komputer), gaya drag sering **disederhanakan** menjadi model linier:

$$\frac{dv}{dt} = -\mu v$$

Dengan μ = koefisien friksi/drag (frictionCoef).
μ juga dapat memakili massa.

Hasil persamaan diferensial di atas adalah:

$$v(t) = v_0 e^{-\mu t}$$

Dengan menurunkan rumus

$$\frac{dv}{dt} = -\mu v$$

kita tahu dengan pasti bagaimana kecepatan berubah seiring waktu secara eksponensial akibat gaya gesekan linier, sehingga bisa digunakan secara akurat dalam simulasi atau analisis fisika.

Tanpa penurunan, kita hanya punya persamaan diferensial tanpa gambaran bentuk solusi kecepatan terhadap waktu.

Bentuk kontinu ke bentuk diskrit (numerik)
Karena pada game waktu tidak lah kontinu karena setiap frame di hitung dalam langkah waktu kecil Δt. Jadi dibutuhkan bentuk diskrit dari solusi di atas:

$$v_{t+\Delta t} = v_t e^{-\mu \Delta t}$$

Dimana $v_{t+\Delta t} = v_t$ adalah vx pada bola, dan di dapat dari perhitungan GLBB.

Jadi kita definisikan

$$drag\_factor = e^{-\mu \Delta t}$$

Maka tiap frame:

$$v \leftarrow v \cdot drag\_factor$$

Penerapan pada kode ditampilkan pada gambar Fig. 10.

7. Pantulan
Ketika bola menabrak permukaan dinding, kecepatan bola berubah arah akibat adanya gaya reaksi dari permukaan.
Namun, tidak semua energi kinetik kembali utuh, sebagian hilang karena gesekan atau pantulan.
Untuk mengukur seberapa "elastis" tabrakan itu, digunakan koefisien restitusi.

Koefisien restitusi, disimbolkan dengan e, adalah ukuran seberapa elastis suatu tumbukan yaitu seberapa besar kecepatan benda setelah memantul dibanding sebelum menumbuk.[8]

$$e = \frac{v_{2f} - v_{1f}}{v_{2i} - v_{2i}}$$

Arti:
e : **Koefisien restitusi** — ukuran seberapa elastis suatu tumbukan
v_{1i}: Kecepatan benda 1 **sebelum** tumbukan
v_{2i}: Kecepatan benda 2 **sebelum** tumbukan
v_{1f}: Kecepatan benda 1 **setelah** tumbukan
v_{2f}: Kecepatan benda 1 **setelah** tumbukan

Karena dinding diam maka: ($v_{2i} = v_{2f} = 0$) jadi rumus dapat disederhanakan:

$$e = \frac{|v_{setelah}|}{|v_{sesudah}|}$$

Artinya:

- Kecepatan setelah pantulan sebanding dengan kecepatan sebelum tumbukan dikalikan e
- Arah kecepatan berbalik, karena benda memantul ke arah berlawanan

Setelah bola menabrak permukaan (misal dinding vertikal), maka:

- Komponen kecepatan **sejajar dinding** tetap (tidak berubah)
- Komponen kecepatan **tegak lurus dinding** berubah arah dan nilainya dikalikan dengan $-e$

Secara matematis:

$$v' = -ev$$

Dimana v adalah kecepatan sebelum tumbukan dan v' adalah kecepatan akhir setelah tumbukan. Penerapan dalam kode di tulis pada Fig. 12.

Dalam implementasi program, nilai **restitution** digunakan sebagai konstanta yang merepresentasikan seberapa elastis tumbukan antara objek dan permukaan. Nilai ini tidak dihitung secara otomatis dari kecepatan sebelum dan sesudah tumbukan, tetapi diberikan secara manual agar simulasi terlihat realistis. Rumus yang digunakan tetap mengikuti persamaan umum fisika $v' = -e\,v$, di mana e adalah koefisien restitusi dan v'merupakan kecepatan setelah tumbukan.

Contoh material dan rentan e di tujukkan pada tabel dibawah ini:

Tabel 2. Material koefisiensi gaya gesek

Material	Rentang (e)	Karakteristik
Bola baja di lantai baja	0.9 – 1.0	Pantulan sangat kuat (hampir elastis sempurna)
Bola tenis / karet	0.7 – 0.9	Pantulan sedang
Kayu	0.4 – 0.6	Pantulan rendah
Lumpur / kain	0.0 – 0.2	Hampir tidak memantul

Karena pada game mirip seperti bola tenis atau karet maka saya menggunakan nilai $e = 0.9$

IV. KESIMPULAN

Penelitian ini berhasil mengembangkan sebuah simulasi dua dimensi (Collision Circle Bounce) yang memodelkan dinamika bola berdasarkan Hukum Newton II dan konsep GLBB, serta mengintegrasikan komponen fisika penting seperti deteksi tumbukan lingkaran, normalisasi vektor, gaya gesek (papan dan udara), dan pantulan dengan koefisien restitusi. Implementasi menggunakan JavaScript dan HTML5 Canvas menunjukkan bahwa model numerik dan logika permainan bekerja sesuai dengan prinsip fisika yang diterapkan setelah dilakukan kalibrasi skala digital—meskipun diperlukan penyesuaian nilai agar perilaku pada domain piksel/frame merefleksikan ekspektasi teori dalam satuan SI.

Kontribusi utama dari proyek ini adalah (1) penyajian model simulasi fisika 2D yang memfokuskan pada dinamika bola sesuai dengan Hukum Newton II dan GLBB, (2) integrasi rumus-rumus fisika nyata (collision, normalisasi, overlap, friction, drag, restitution) langsung ke dalam implementasi kode, dan (3) penyediaan media interaktif berbasis pemrograman yang dapat mendukung pembelajaran konsep fisika dasar sekaligus melatih keterampilan logika pemrograman. Hasil pengujian pada beberapa skenario (dorongan tunggal, variasi koefisien restitusi, pengaruh gesekan, dan interaksi bot) menunjukkan konsistensi perilaku simulasi dengan ekspektasi teori dalam konteks pendidikan/visualisasi.

Keterbatasan yang perlu diperhatikan adalah penggunaan integrator Euler diskrit yang berpotensi menghasilkan error numerik untuk langkah waktu (dt) yang besar, serta asumsi penyederhanaan (mis. tidak memodelkan rotasi/spin bola, deformasi material, atau turbulensi) yang membuat simulasi lebih cocok sebagai alat pembelajaran daripada pengganti eksperimen fisik. Untuk pengembangan lanjutan disarankan peningkatan akurasi numerik (mis. integrator Runge–Kutta), penambahan efek rotasi dan torsi, model drag yang lebih realistis, serta validasi eksperimen dengan data fisik untuk memperkuat kesesuaian kuantitatif antara simulasi dan dunia nyata.

PENGAKUAN

Penulis mengucapkan terima kasih kepada:

Fakultas Teknologi dan Rekayasa, Program Studi Teknik Robotika dan Kecerdasan Buatan, Universitas Sunan Gresik atas dukungan fasilitas akademik dan kesempatan pelaksanaan proyek ini.

Dosen pengampu mata kuliah Fisika Dasar dan rekan-rekan yang memberikan masukan, saran, serta berperan sebagai penguji dan pengamat selama tahap pengujian.

Pengembang dan penyedia alat kerja yang digunakan dalam proyek ini (termasuk Visual Studio Code, web browser, serta teknologi HTML5 Canvas dan JavaScript) yang memfasilitasi implementasi dan visualisasi.

Sumber daya dan literatur yang menjadi rujukan teknis, serta alat bantu AI yang memberi saran teknis untuk proses kalibrasi nilai digital selama pengembangan.

Dukungan dalam bentuk masukan teknis, fasilitas pengujian, dan umpan balik pengguna sangat berperan dalam terselesainya penelitian ini.

REFERENSI

[1] **Circle-Circle Collision Detection** 2015-2016, Kishimoto Studios

[2] Air Friction HyperPhysics@2024C.R. Nave Gerogia State University

[3] David Baraff – Physically Based Modeling (SIGGRAPH course notes)

[4] Witkin & Baraff – Physically Based Modeling: Numerical Integration

[5] R. A. Serway and J. W. Jewett, **Physics for Scientists and Engineers with Modern Physics**, 9th ed. Boston, MA: Cengage Learning, 2014, part I, ch 3, pp. 59–65.

[6] Millington, I. (2010). Game Physics Engine Development. Second Edition. Morgan Kaufmann. Ch 14.2, pp 306-320

[7] **Giancoli, D. C. (2014). *Physics: Principles with Applications* (2nd ed., Ch 2 & Ch 4, pp. 78). Pearson**

[8] Two-Body Collisions and the Coefficient of Restitution, Univ. of Florida