# LAPORAN PENGERJAAN TUGAS PRAKTIKUM
# MATA KULIAH KOMPUTER GRAFIK

## Tugas Week 3 Core Mechanics - Shooting, UI, dan Game State

Disusun untuk memenuhi tugas Komputer Grafik


**Disusun oleh :**


| | | |
|---|---|---|
| Nama | : | Muhammad Samudera Bagja |
| NIM | : | 231524058 |
| Kelas | : | D4 – 2B Teknik Informatika |

**PROGRAM STUDI SARJANA TERAPAN TEKNIK INFORMATIKA**

**JURUSAN TEKNIK INFORMATIKA**

**POLITEKNIK NEGERI BANDUNG**


**TAHUN AJARAN 2024/2025**

# DAFTAR ISI

# Lesson Learned dari Penugasan Week 3: Core Mechanics - Shooting, UI, dan Game State

1. **Penggunaan Collision Layers & Masks**

- **Apa yang dipelajari:**
    - Collision Layers & Mask memungkinkan kontrol yang presisi terhadap interaksi fisik antar objek.
    - **Layer** menentukan kategori fisik objek,
      sedangkan **Mask** menentukan layer mana yang akan dideteksi oleh objek tersebut.
    - Interaksi hanya terjadi jika kedua objek saling memenuhi syarat Layer dan Mask.
- **Kesalahan yang mungkin terjadi:**
    - Lupa mengatur Layer dan Mask dengan benar, menyebabkan objek tidak saling mendeteksi atau bertabrakan.
    - Tidak memberikan nama yang jelas pada Layer di **Project Settings**, sehingga sulit dikelola.
- **Solusi & Best Practice:**
    - Selalu beri nama yang deskriptif pada Layer
      (misal: player, enemy, bullet).
    - Verifikasi interaksi dengan testing manual.

---

2. **Pemanfaatan Signals & Area3D**

- **Apa yang dipelajari:**
    - **Signals** memungkinkan komunikasi antar node tanpa hard dependency.
    - **Area3D** berguna untuk deteksi non-fisik (misal: trigger zone).
    - Signal body_entered dan area_entered dapat dihubungkan via Editor atau kode.
- **Kesalahan yang mungkin terjadi:**
    - Signal tidak terhubung karena salah memilih target method.
    - Lupa memeriksa body yang masuk (if (body == this)) sehingga logika terpicu oleh objek yang salah.
- **Solusi & Best Practice:**

- Gunakan **Debug -> Visible Collision Shapes** untuk memvisualisasikan Area3D.
- Selalu validasi objek yang memicu signal.

3. **Dynamic Object Spawning dengan Scene Instancing**

- **Apa yang dipelajari:**
  - Objek dapat di-spawn secara dinamis menggunakan PackedScene.Instantiate().
  - Instance baru harus ditambahkan ke scene tree dengan AddChild().
  - QueueFree() digunakan untuk menghapus objek dengan aman.
- **Kesalahan yang mungkin terjadi:**
  - Lupa menambahkan instance ke scene tree, menyebabkan objek tidak muncul.
  - Salah mengatur transformasi (posisi/rotasi) objek yang di-spawn.
- **Solusi & Best Practice:**
  - Pastikan path PackedScene benar dan di-load sebelum runtime.
  - Gunakan GlobalTransform untuk mengatur posisi/rotasi relatif terhadap dunia.

**4. Implementasi UI dengan CanvasLayer & Label**

- **Apa yang dipelajari:**
  - **CanvasLayer** memisahkan UI dari dunia game dan tidak terpengaruh kamera.
  - **Label** dapat diupdate via script dengan mengubah properti Text.
  - GameManager berguna sebagai pusat kontrol game state (misal: score).
- **Kesalahan yang mungkin terjadi:**
  - UI tidak muncul karena salah mengatur **Layout** atau **Z Index**.
  - Salah path saat mengambil referensi Label dari kode.
- **Solusi & Best Practice:**
  - Gunakan **Theme Overrides** untuk mengatur ukuran/warna font.

- Simpan referensi UI node di _Ready() untuk menghindari GetNode berulang.

---

**5. Manajemen Node dengan Groups**

- **Apa yang dipelajari:**
  - **Groups** memudahkan pengelolaan node terkait (misal: semua musuh).
  - GetTree().GetNodesInGroup() mengembalikan array node dalam group.
- **Kesalahan yang mungkin terjadi:**
  - Lupa menambahkan node ke group.
  - Tidak memeriksa tipe node sebelum memanggil method spesifik.
- **Solusi & Best Practice:**
  - Selalu lakukan pengecekan tipe (if (node is RigidBody3D)) sebelum operasi.
  - Gunakan Group untuk broadcast event (misal: reset semua musuh).

---

**6. Game State Management**

- **Apa yang dipelajari:**
  - Variabel seperti score dan isGameOver mengontrol alur game.
  - Win/Lose condition dapat diimplementasikan dengan pengecekan sederhana.
- **Kesalahan yang mungkin terjadi:**
  - Game state tidak di-reset saat restart.
  - Race condition jika multiple node mengubah state bersamaan.
- **Solusi & Best Practice:**
  - Gunakan **Singleton Pattern** (Autoload) untuk GameManager jika perlu akses global.
  - Hindari modifikasi state langsung dari banyak script.

---

**7. Pembuatan Game "Target Practice" (Assignment)**

- **Apa yang dipelajari:**

    - Menggabungkan semua konsep: shooting, UI, collision, dan game state.
    - **RigidBody3D** bisa digunakan untuk target yang jatuh saat terkena tembakan.
- **Kesalahan yang mungkin terjadi:**
    - Bullet tidak menghancurkan target karena salah Layer/Mask.
    - Score tidak terupdate karena salah pemanggilan GameManager.
- **Solusi & Best Practice:**
    - Gunakan **Debug Print** untuk memverifikasi signal dan method terpanggil.
    - Implementasikan **object pooling** jika banyak bullet di-spawn (optimasi).

---

**Kesimpulan**

- **Collision Layers/Mask** dan **Signals** adalah fondasi interaksi game.
- **Scene Instancing** penting untuk mekanik dinamis (bullet, enemy spawn).
- **GameManager** + **UI** memberikan feedback jelas ke pemain.
- **Testing berulang** sangat penting untuk memastikan semua sistem bekerja.

Dengan memahami konsep ini, game mechanics yang lebih kompleks (seperti enemy AI, power-ups, atau level progression) bisa dikembangkan lebih mudah.