

**LAPORAN**  
**UJIAN AKHIR SEMESTER**  
**STUDI KASUS: GAME EDUKASI MENGETIK**  
**“CYBER TYPER: NEON PROTOCOL”**

**Mata Kuliah:**

Pemrograman Berorientasi Objek



**Dosen Pengampu:**  
M Adamu Islam Mashuri, S.Tr.T., M.Tr.Kom.

**Dibuat Oleh:**

SELVI ADINDA HERMAWATI

24091397145

2024E

**PROGRAM STUDI**  
**D4 MANAJEMEN INFORMATIKA**  
**FAKULTAS VOKASI**  
**UNIVERSITAS NEGERI SURABAYA**  
**2025**

## **BAB 1**

### **PENDAHULUAN**

#### **1.1 PENDAHULUAN DAN LATAR BELAKANG PROYEK**

“**Cyber Typer: Neon Protocol**” adalah sebuah game edukatif yang dirancang sebagai media pembelajaran untuk meningkatkan kemampuan mengetik secara cepat dan akurat. Permainan ini memadukan elemen hiburan interaktif dengan tujuan edukatif, sehingga proses latihan mengetik menjadi lebih menarik dan tidak membosankan. Pada era digital saat ini, keterampilan mengetik merupakan kemampuan dasar yang sangat dibutuhkan dalam kegiatan akademik maupun profesional, sehingga diperlukan media latihan yang efektif dan mudah digunakan.

Pengembangan game ini dipilih karena selain memiliki nilai edukatif, karakteristik game mengetik juga sangat sesuai untuk penerapan konsep Pemrograman Berorientasi Objek (PBO/OOP). Proses permainan menuntut pengelolaan berbagai objek secara bersamaan, pemrosesan input pengguna secara real-time, serta pengaturan data permainan seperti skor dan nyawa. Seluruh kebutuhan tersebut dapat diimplementasikan secara terstruktur melalui penerapan konsep OOP, termasuk encapsulation, inheritance, polymorphism, dan abstraction.

“**Cyber Typer: Neon Protocol**” dikembangkan dengan memanfaatkan struktur kelas yang modular, sehingga setiap komponen memiliki tanggung jawab yang jelas. Mulai dari pengelolaan alur permainan, penyimpanan data, hingga penggambaran objek visual seperti meteor, partikel, dan teks efek. Dengan konsep tersebut, game ini tidak hanya menjadi media hiburan, tetapi juga menjadi contoh nyata bagaimana OOP dapat diterapkan dalam proyek pengembangan aplikasi berskala menengah.

Proyek ini juga menjadi sarana bagi pengembang (mahasiswa) untuk memperoleh pengalaman langsung dalam merancang arsitektur perangkat lunak, memahami hubungan antar objek, serta mengimplementasikan logika permainan yang responsif. Dengan demikian, hasil pengembangan game ini diharapkan dapat memberikan manfaat pedagogis, baik dari sisi desain perangkat lunak maupun dari sisi pengguna akhir.

#### **1.2 TUJUAN**

- a) Mengembangkan game edukatif “**Cyber Typer: Neon Protocol**” sebagai media pembelajaran untuk melatih kecepatan dan akurasi mengetik.
- b) Menerapkan konsep Pemrograman Berorientasi Objek (PBO/OOP) dalam pengembangan game, meliputi pengelolaan objek, alur permainan, dan data pengguna.
- c) Memahami dan melatih perancangan arsitektur perangkat lunak berbasis OOP melalui implementasi game dengan struktur kelas yang modular dan terstruktur.

## **BAB 2**

### **LANDASAN TEORI**

#### **2.1 PEMROGRAMAN BERBASIS OBJEK (PBO)**

Pemrograman berorientasi objek (Inggris: Object-oriented programming disingkat OOP) merupakan paradigma pemrograman berdasarkan konsep “objek”, yang dapat berisi data, dalam bentuk field atau dikenal juga sebagai atribut; Serta kode, dalam bentuk fungsi/prosedur yang dikenal juga sebagai method. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya.

Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam Teknik peranti lunak skala besar. Lebih jauh lagi pendukung OOP mengklaim bahwa OOP lebih mudah dipelajari bagi pemula dibanding dengan pendekatan sebelumnya, dan pendekatan OOP lebih mudah dikembangkan dan dirawat.

#### **2.2 KONSEP DASAR OOP**

##### **a) Encapsulation**

Enkapsulasi adalah proses penyembunyian detail implementasi suatu objek dari dunia luar. Hanya *interface* publik (seperti *method* yang disediakan objek) yang bisa diakses, sementara data internal dilindungi agar tidak sembarangan diubah.

##### **b) Inheritance**

Inheritance memungkinkan sebuah *class* (kelas) untuk mewarisi properti dan *method* dari *class* lain. Ini memungkinkan pengkodean yang lebih efisien dan menjadikan OOP sangat cocok untuk membangun aplikasi skala besar.

##### **c) Polymorphism**

Polimorfisme memungkinkan objek untuk memiliki banyak bentuk misalnya, *method* dengan nama sama bisa berperilaku berbeda tergantung pada konteksnya. Ini meningkatkan fleksibilitas dan dinamis dalam pengkodean.

#### **d) Abstraction**

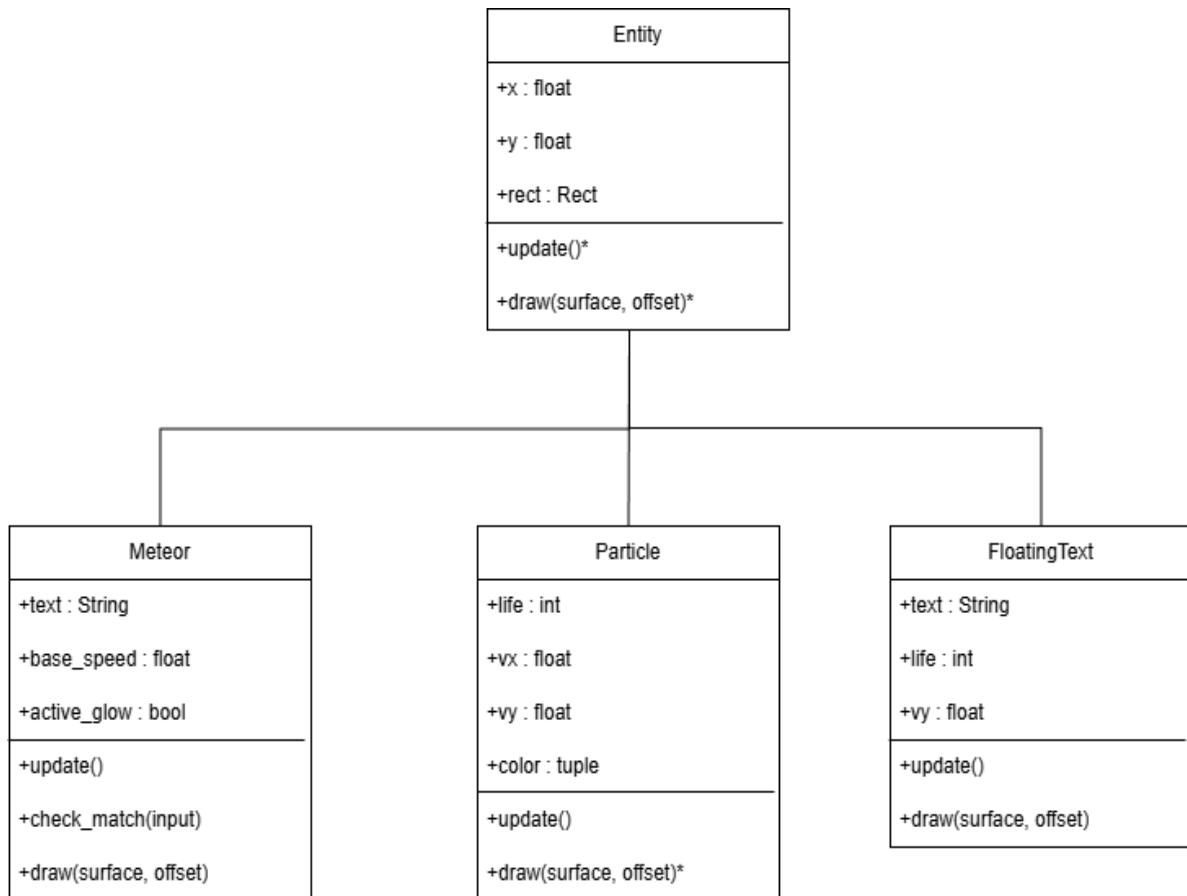
Abstraksi memungkinkan kita untuk menyembunyikan kompleksitas dan hanya menampilkan fitur penting dari suatu objek. Misalnya, saat menggunakan kendaraan, kamu cukup tahu cara menyalakannya tanpa harus tahu mekanisme mesin di dalamnya.

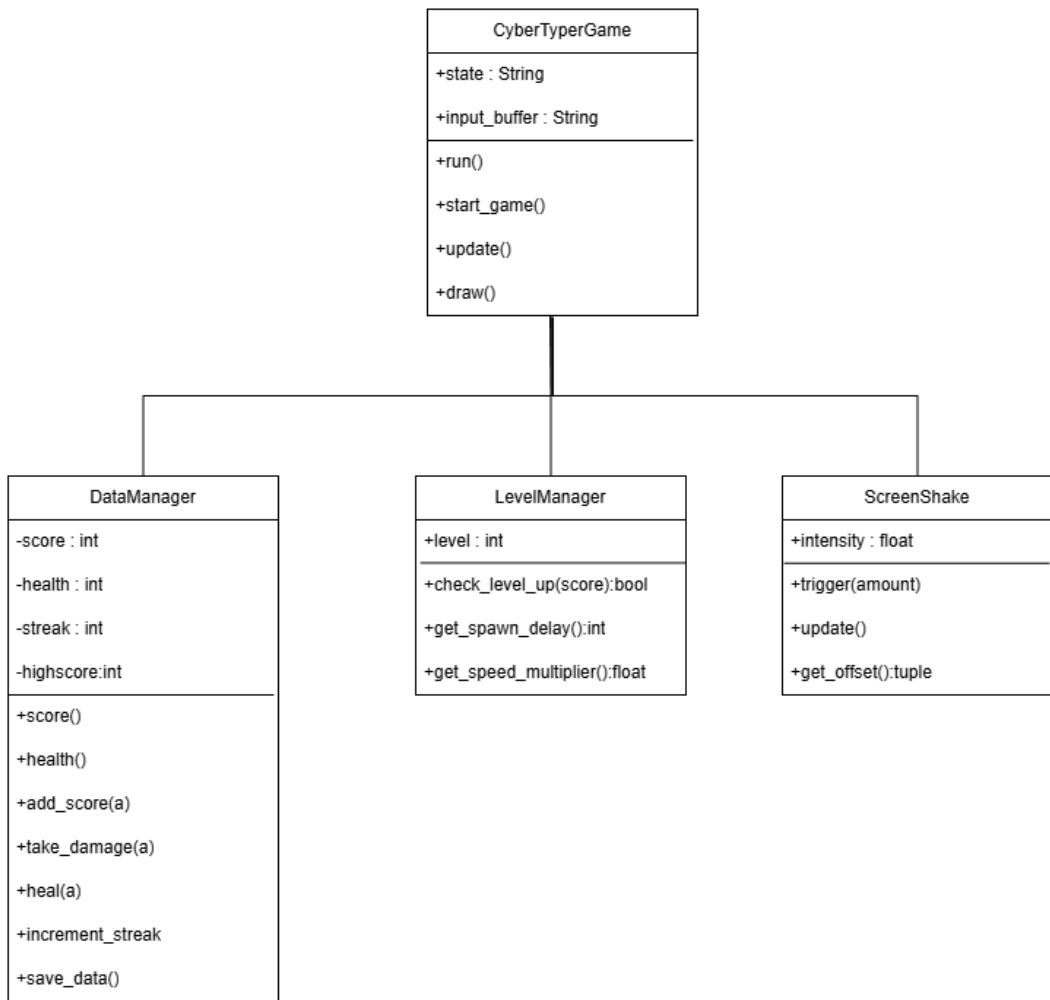
## BAB III

### PERANCANGAN SISTEM

#### CLASS DIAGRAM

Sesuai dengan persyaratan dokumentasi, berikut adalah rancangan class diagram dibuat untuk menggambarkan hubungan antar komponen (kelas) dalam aplikasi. Diagram dibawah ini juga memastikan bahwa pengembangan berjalan secara terstruktur dan sesuai dengan prinsip *Separation of Concerns*.

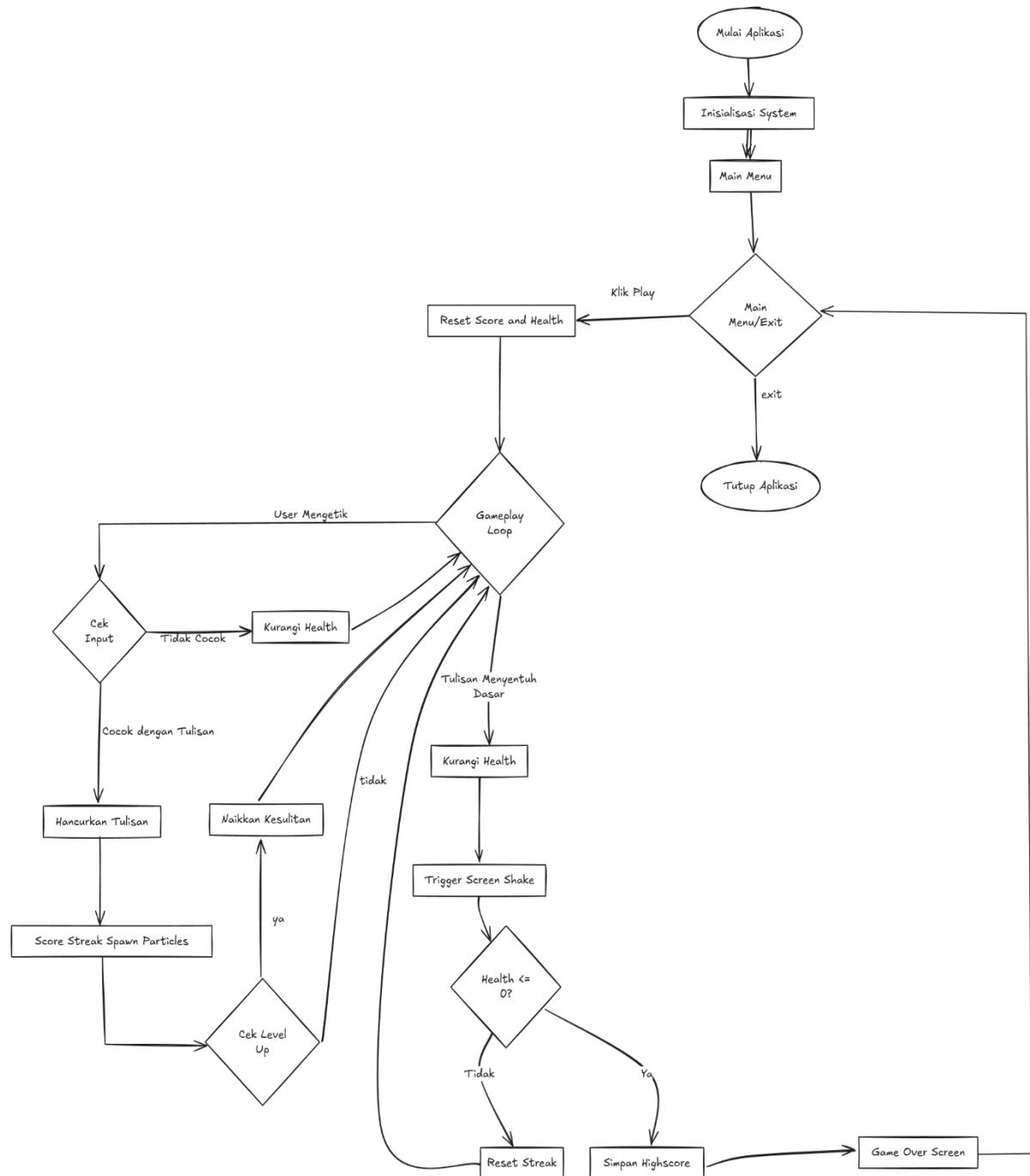




Berdasarkan diagram kelas yang dirancang, sistem dibangun menggunakan struktur modular untuk memisahkan tanggung jawab ( Separation of Concerns):

- CyberTyperGame (Controller): Bertindak sebagai otak utama aplikasi atau bisa disebut sebagai pusat kontrol utama aplikasi. Kelas ini mengelola alur permainan mulai dari inisialisasi, loop permainan utama, hingga transisi antar state (Menu, Playing, Game Over).
- DataManager (Model): Kelas ini didedikasikan khusus untuk menangani data statistik pemain. Ia menyimpan status kritis seperti skor, nyawa (health), dan streak kemenangan. Data ini dilindungi dengan metode OOP untuk menjaga integritasnya.
- Entity (Abstract Base Class): Merupakan kelas induk abstrak yang menjadi cetak biru bagi seluruh objek visual dalam game. Kelas ini memastikan bahwa setiap objek memiliki properti dasar koordinat (X, Y) dan kemampuan untuk digambar ke layar.
- Meteor, Particle, & FloatingText (View/Object): Merupakan kelas turunan yang

mewarisi sifat Entity. Masing-masing memiliki perilaku unik: Meteor jatuh ke bawah, Partikel menyebar saat ledakan, dan Teks melayang(floating text) memberikan umpan balik visual.



## **Alur Aplikasi (Flowchart)**

Alur permainan pada aplikasi **Cyber Typer: Neon Protocol** dirancang secara sistematis dan terstruktur untuk memastikan proses permainan berjalan dengan lancar, interaktif, serta mudah dipahami oleh pengguna. Berikut adalah penjelasan detail dari setiap tahapan alur aplikasi:

1. **Inisialisasi:** Pada tahap awal, ketika aplikasi dijalankan (**Start / Mulai Aplikasi**), sistem melakukan proses inisialisasi. Proses ini mencakup pemuatan seluruh komponen penting yang dibutuhkan selama permainan berlangsung, seperti pengaturan layar (window), suara, asset visual, serta variabel utama permainan. Selain itu, sistem juga memuat data **Highscore** yang tersimpan dalam penyimpanan eksternal berupa file **JSON**. Data ini digunakan untuk menampilkan dan membandingkan skor tertinggi yang pernah dicapai oleh pemain sebelumnya.
2. **Menu:**

Setelah proses inisialisasi selesai, aplikasi akan menampilkan Menu Utama (Main Menu). Pada menu ini, pengguna diberikan beberapa pilihan, seperti:

  - Play / Start untuk memulai permainan
  - Exit untuk keluar dari aplikasi

Apabila pengguna memilih tombol Exit, sistem akan langsung menghentikan proses dan menutup aplikasi. Sebaliknya, jika pengguna memilih tombol Play, sistem akan melanjutkan ke tahap berikutnya, yaitu memulai permainan.
3. **Gameplay Loop:**
  - Sistem memunculkan objek kata (Meteor).
  - Pengguna memberikan input ketikan. Sistem memvalidasi input tersebut.
  - **Benar:** Skor +, Streak +, Muncul Partikel.
  - **Salah:** Nyawa berkurang (via Encapsulation).
4. **Game Over:** Jika nyawa mencapai 0, permainan masuk ke status *Game Over*, skor tersimpan jika memecahkan rekor, dan kembali ke menu utama.

## BAB IV

### IMPLEMENTASI KONSEP OOP

#### 1. Encapsulation (Enkapsulasi) untuk Keamanan Data

Masalah utama dalam pengembangan game adalah kerentanan variabel global yang dapat diubah secara tidak sengaja oleh bagian kode lain

- **Masalah:** Data statistik pemain (Skor, Nyawa, Streak) rentan dimanipulasi secara tidak sengaja oleh bagian kode lain.
- **Solusi:** Menggunakan *Access Modifier* private (simbol `_`) pada kelas DataManager. Akses data hanya bisa dilakukan melalui metode *getter/setter* yang tervalidasi.
- **Mekanisme Akses:** Akses ke data ini tidak dilakukan secara langsung, melainkan melalui metode *getter* (menggunakan dekorator `@property`) dan metode modifikasi yang tervalidasi seperti `take_damage()`
- **Manfaat:** Hal ini mencegah manipulasi nilai yang tidak valid (misalnya nyawa menjadi negatif tanpa melalui logika pengurangan yang benar) dan menjaga integritas data permainan.

Implementasi Kode:

```
class DataManager:  
    def __init__(self):  
        self.__score = 0  
        self.__health = 100  
        self.__streak = 0  
  
    @property  
    def health(self): # Getter  
        return self.__health  
  
    def take_damage(self, amount):  
        self.__health -= amount  
        self.reset_streak()
```

Bukti Tampilan Aplikasi:



Tampilan UI Health dan Score yang datanya dilindungi Encapsulation.

## 2. Inheritance (Pewarisan)

Tanpa pewarisan, pengembang harus menulis ulang kode inisialisasi posisi X dan Y untuk setiap objek visual (Meteor, Partikel, Teks), yang menyebabkan duplikasi kode (*redundancy*)

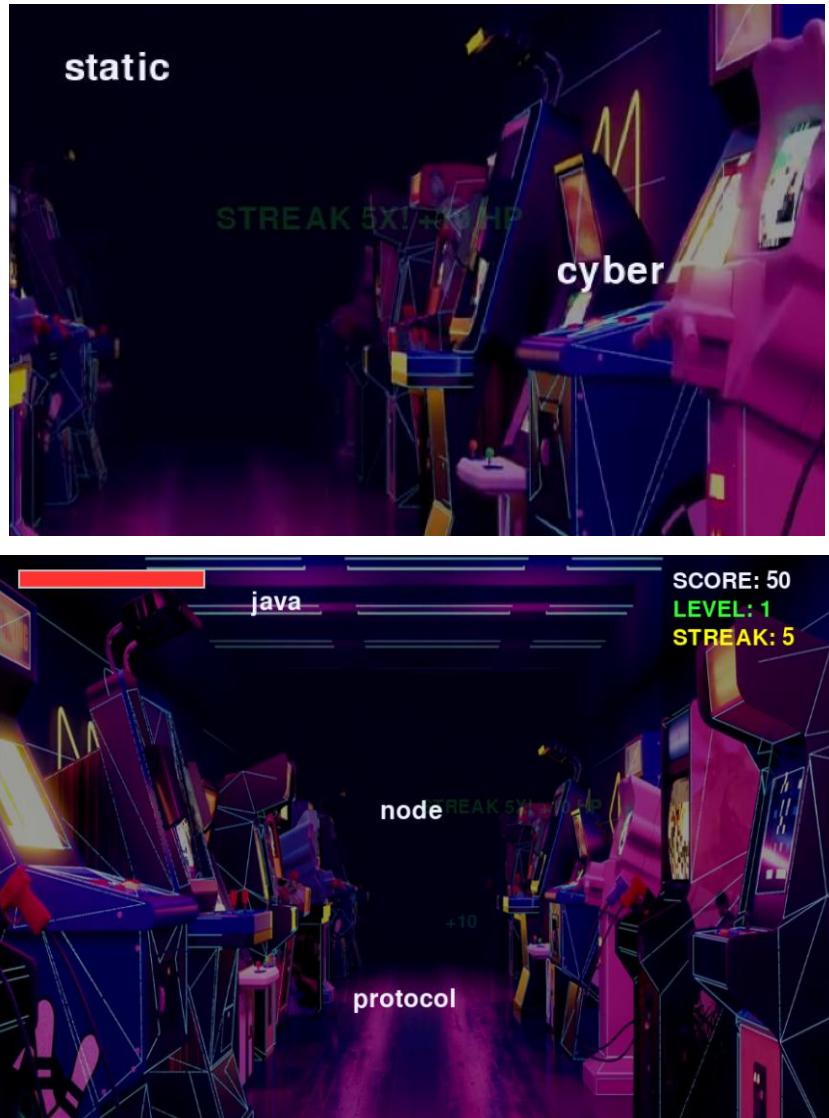
- **Masalah:** Terdapat banyak objek visual (Meteor, Partikel, Teks Melayang) yang memiliki properti sama (koordinat X, Y) tetapi ditulis berulang-ulang.
- **Solusi:** Membuat kelas induk Entity yang mewariskan properti dasar ke semua objek visual.
- **Penerapan:** Kelas Meteor cukup mewarisi Entity (class Meteor(Entity)), sehingga otomatis memiliki properti koordinat tanpa perlu mendefinisikannya ulang. Kode pada kelas anak hanya fokus pada logika spesifik, seperti kecepatan jatuh meteor atau teks yang dibawa .
- **Dampak:** Mengurangi penulisan kode berulang hingga diperkirakan 40%, membuat kode lebih bersih dan mudah dikelola.

```
class Entity(ABC, pygame.sprite.Sprite):
    def __init__(self, x, y):
        super().__init__()
        self.x = x
        self.y = y

class Meteor(Entity):
    def __init__(self, text, speed):
        super().__init__(random.randint(50, 800), -60)
        self.text = text
```

Bukti Tampilan Aplikasi:





Meteor dan Partikel adalah sebuah objek berbeda yang mewarisi kelas induk Entity yang sama.

### 3. Polymorphism (Polimorfisme)

Tantangan teknis muncul ketika sistem harus menggerakkan berbagai jenis benda yang memiliki pola gerakan berbeda (Meteor bergerak vertikal, Partikel bergerak menyebar/radial).

- **Masalah:** Sistem utama harus menggerakkan semua benda. Meteor bergerak ke bawah, sedangkan Partikel bergerak menyebar. Menggunakan if-else untuk mengecek tipe benda tidak efisien.
- **Solusi:** Menggunakan *Method Overriding*. Semua kelas anak memiliki metode update() yang sama namanya, tapi beda isinya.
- **Meteor:** Metode update() berisi logika self.y += self.base\_speed (gerak vertikal).

- **Particle:** Metode update() berisi logika self.x += self.vx dan self.y += self.vy (gerak menyebar) .
- **Game Loop:** Sistem utama tidak perlu mengecek tipe objek menggunakan if-else. Cukup memanggil obj.update() dalam satu baris perintah untuk seluruh objek, dan setiap objek akan bergerak sesuai sifatnya masing-masing.

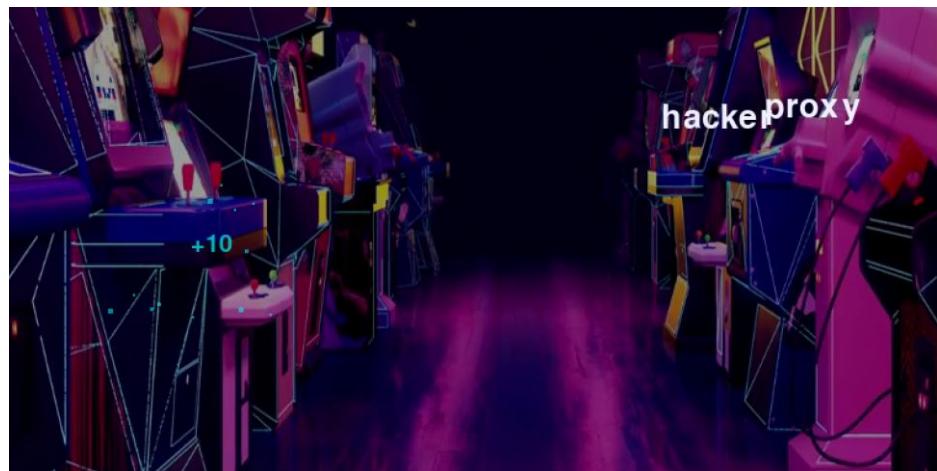
### Implementasi Kode:

```
def update(self):
    self.y += self.base_speed

# Pada Kelas Particle
def update(self):
    self.x += self.vx
    self.y += self.vy

for obj in all_objects:
    obj.update()
```

Bukti Tampilan Aplikasi:



Pada gambar diatas Partikel menyebar sementara Meteor tetap jatuh lurus.

## BAB V

### FITUR KREATIVITAS & TAMBAHAN

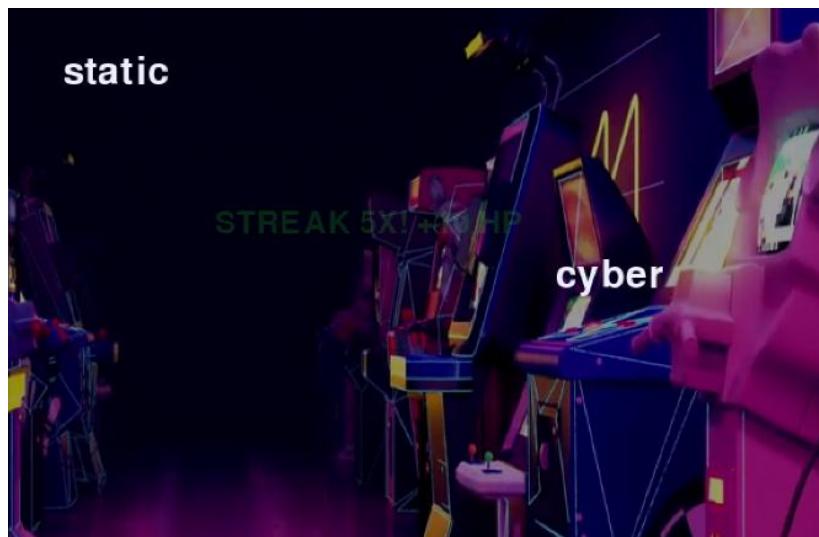
Aplikasi ini mengintegrasikan fitur tambahan untuk meningkatkan pengalaman pengguna (UX).

**A. Sistem Streak & Reward (Risk vs Reward)** Pemain mendapat bonus Heal (+HP) jika berhasil mengetik 5 kata berturut-turut. Logika ini tersimpan di DataManager.

**Logika:** Menggunakan operasi modulus (if self.\_\_streak % 5 == 0). Setiap pemain berhasil mengetik 5 kata berturut-turut tanpa salah, sistem memberikan bonus penyembuhan (*Heal*)

**Tujuan:** Mendorong pemain untuk mengetik dengan hati-hati (akurasi) bukan hanya sekadar cepat.

```
# Potongan Kode Streak
def increment_streak(self):
    self.__streak += 1
    if self.__streak % 5 == 0:
        return True
```



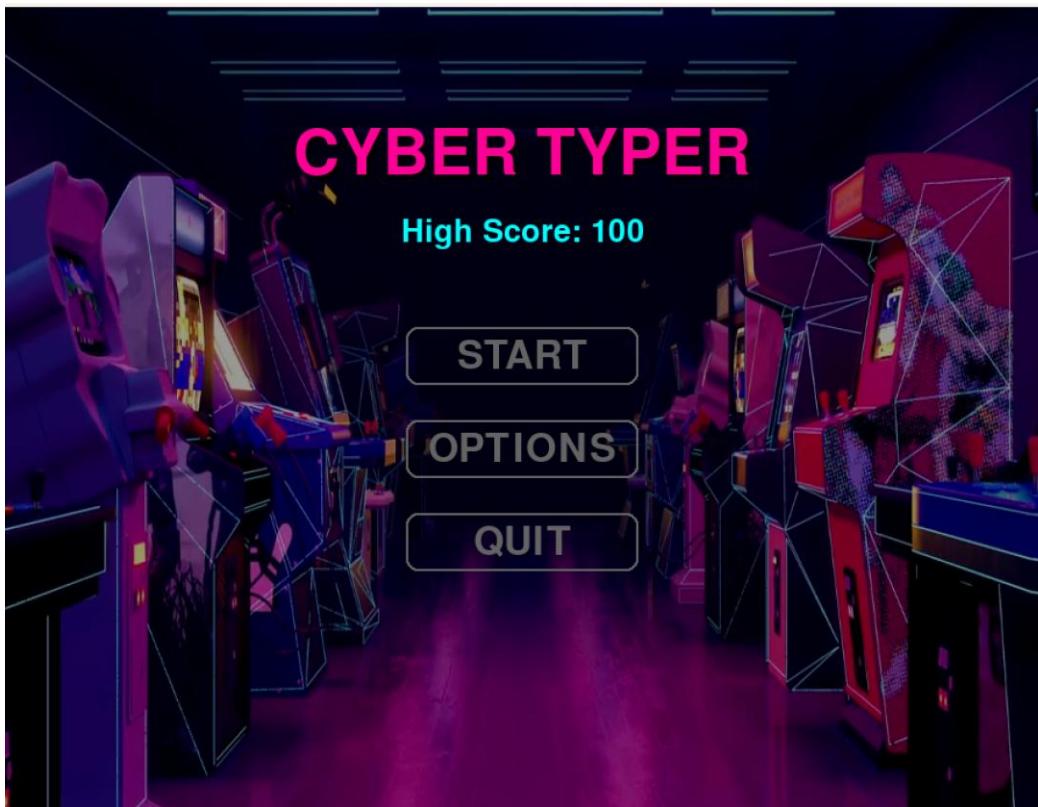
Pada gambar diatas menunjukkan bahwa gambar tersebut adalah sebuah Notifikasi Streak Bonus.

**B. Persistensi Data (JSON Highscore)** Menggunakan modul json untuk menyimpan skor tertinggi secara permanen agar tidak hilang saat aplikasi ditutup.

- **Implementasi:** Fungsi `save_data` mengecek apakah skor saat ini lebih tinggi dari `highscore` yang ada. Jika ya, data ditulis ke file `game_data.json`

- **Manfaat:** Skor tertinggi tidak akan hilang meskipun komputer dimatikan atau aplikasi ditutup, memberikan nilai kompetitif jangka panjang bagi pemain.

```
def save_data(self):
    if self.__score > self.__highscore:
        with open("game_data.json", "w") as f:
            json.dump({"highscore": self.__score}, f)
```



High Score yang tersimpan permanen di Menu Utama.

## **BAB VI**

### **PENUTUP**

#### **5.1 KESIMPULAN**

Pengembangan "*Cyber Typer: Neon Protocol*" membuktikan bahwa penerapan OOP sangat baik dalam pengembangan game modern.

1. **Keamanan:** Enkapsulasi melindungi data vital dari *bug* logika.
2. **Efisiensi:** Pewarisan memangkas duplikasi kode secara signifikan.
3. **Fleksibilitas:** Polimorfisme memungkinkan penambahan fitur baru (misalnya tipe musuh baru) di masa depan tanpa merombak kode utama.
4. **Kreativitas:** Integrasi fitur seperti JSON dan sistem *Streak* menjadikan game ini tidak hanya fungsional secara teknis, tetapi juga menarik dari sisi pengalaman pengguna..

#### **5.2 SARAN**

- Game "*Cyber Typer: Neon Protocol*" dapat dikembangkan lebih lanjut dengan menambahkan variasi level dan mode permainan agar tidak monoton.
- Tampilan antarmuka dan efek visual dapat disempurnakan untuk meningkatkan kenyamanan dan pengalaman pengguna.
- Penerapan konsep OOP dapat diperluas agar struktur kode semakin fleksibel dan mudah dikembangkan di masa depan.

#### **5.3 LAMPIRAN**

- **Link Github**

<https://github.com/ssyhadins8/-CYBER-TYPER-NEON-PROTOCOL-/tree/main>