Implementasi Object-Oriented Programming dalam Pengembangan Game Edukatif Polymor-Farm

Mata Kuliah:

Pemrograman Berorientasi Objek

Disusun Oleh:

Nayla Rahayu Puspitasari (24091397088)

UNIVERSITAS NEGERI SURABAYA

FAKULTAS VOKASI

PROGRAM STUDI D4 MANAJEMEN INFORMATIKA

2025

DAFTAR ISI

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1. Latar Belakang

Pendidikan jenjang sekolah dasar memiliki peran penting dalam membentuk kemampuan kognitif, afektif, dan psikomotorik pada anak. Anak usia sekolah dasar berada pada tahap perkembangan kognitif, yaitu fase di mana anak lebih mudah memahami konsep melalui pengalaman langsung dan visual dibandingkan penjelasan abstrak semata. Oleh karena itu, media pembelajaran yang interaktif dan berbasis pengalaman sangat dianjurkan untuk mendukung proses belajar yang optimal.

Pemanfaatan teknologi dalam pendidikan dasar mampu menjadi sarana pendukung pembelajaran yang efektif dan menarik. *Game* edukatif merupakan salah satu media yang berpotensi besar karena mampu menggabungkan unsur hiburan dan pembelajaran (*edutainment*). *Game-based learning* dapat meningkatkan motivasi, keterlibatan, dan pemahaman siswa karena anak belajar melalui aktivitas yang mereka sukai. Dengan demikian, *game* edukatif dapat membantu anak memahami konsep dasar seperti tanggung jawab, logika sebab–akibat, serta kemampuan berhitung secara alami.

Namun, pada masa saat ini sebagian besar *game* yang dimainkan oleh anak sekolah dasar lebih berorientasi pada hiburan semata dan minim nilai edukatif. Banyak *game* tidak dirancang untuk melatih kemampuan berpikir logis atau menanamkan nilai pembelajaran, melainkan hanya menekankan kompetisi, kecepatan, atau aspek visual tanpa makna pendidikan yang jelas. Di sisi lain, media pembelajaran digital yang digunakan di sekolah masih cenderung bersifat pasif, seperti video atau modul interaktif sederhana, sehingga kurang melibatkan anak secara aktif dalam proses belajar.

Berdasarkan permasalahan tersebut, penelitian ini berfokus pada pengembangan sebuah aplikasi *game* edukatif sederhana untuk anak sekolah dasar berjudul Polymor-Farm. *Game* ini dirancang untuk menghadirkan pembelajaran berbasis pengalaman melalui aktivitas merawat hewan ternak, seperti memberi makan, menjaga kebahagiaan, dan mengelola hasil produksi. Konsep perubahan bentuk hewan ketika dirawat dengan baik memberikan umpan balik visual yang jelas, sehingga anak dapat memahami hubungan sebab–akibat secara langsung dari tindakan yang mereka lakukan.

Penelitian ini diharapkan dapat menjadi media pembelajaran yang efektif dalam menanamkan nilai tanggung jawab, kemampuan berhitung sederhana, serta logika dasar pada anak sekolah dasar. Selain itu, penelitian ini juga memberikan kontribusi dalam pengembangan media pembelajaran digital yang lebih sesuai dengan kebutuhan dan karakteristik peserta didik usia dini.

1.2. Rumusan Masalah

Rumusan masalah dalam game Polymor-Farm dapat dijabarkan sebagai berikut:

1. Apa peran game Polymor-Farm sebagai media pembelajaran bagi anak sekolah dasar?
2. Bagaimana kesesuaian game Polymor-Farm dengan konsep dasar Object-Oriented Programming (OOP)?
3. Apa fitur utama dalam game Polymor-Farm yang mendukung proses pembelajaran anak sekolah dasar?

1.3. Tujuan

Tujuan dari ? ini adalah:

1. Mengetahui peran game Polymor-Farm sebagai media pembelajaran bagi anak sekolah dasar.
2. Menganalisis kesesuaian penerapan konsep dasar Object-Oriented Programming (OOP) dalam game Polymor-Farm.
3. Mengidentifikasi fitur-fitur utama dalam game Polymor-Farm yang mendukung proses pembelajaran anak sekolah dasar

BAB II
LANDASAN TEORI

2.1. Game Edukatif

Game edukatif merupakan media pembelajaran digital yang menggabungkan unsur hiburan dan pembelajaran dengan tujuan meningkatkan motivasi serta pemahaman pengguna. *Game* edukatif dirancang tidak hanya untuk memberikan kesenangan, tetapi juga untuk menyampaikan materi pembelajaran secara interaktif dan kontekstual. Pendekatan ini dikenal sebagai *edutainment*, yaitu metode pembelajaran yang mengintegrasikan edukasi dan hiburan.

Dalam konteks pendidikan anak sekolah dasar, *game* edukatif berperan penting karena mampu menyajikan pembelajaran berbasis pengalaman langsung. Anak dapat belajar melalui interaksi, pengambilan keputusan, dan umpan balik visual yang diberikan oleh sistem permainan. Melalui *game* edukatif, konsep-konsep dasar seperti tanggung jawab, logika sebab akibat, serta kemampuan berhitung dapat disampaikan secara alami tanpa memberikan beban kognitif berlebih kepada anak.

Dengan demikian, *game* edukatif dapat dijadikan sebagai media pembelajaran alternatif yang efektif dan menarik, khususnya bagi anak sekolah dasar yang membutuhkan pembelajaran bersifat konkret, visual, dan interaktif.

2.2. Object-Oriented Programming (OOP)

Object Oriented Programming (OOP) merupakan paradigma pemrograman yang berorientasi pada objek, di mana program dibangun berdasarkan kumpulan objek yang memiliki atribut dan metode. Pendekatan ini bertujuan untuk meningkatkan keteraturan kode, kemudahan pemeliharaan, serta memungkinkan pengembangan perangkat lunak yang modular dan dapat digunakan kembali (*reusability*). Hibatullah (2021) menjelaskan bahwa OOP memiliki empat prinsip utama, yaitu *encapsulation*, *inheritance*, *polymorphism*, dan *abstraction*. Pendekatan OOP dapat diterapkan pada aplikasi berbasis visual seperti *game* edukatif karena setiap elemen dalam *game* dapat dimodelkan sebagai objek dengan karakteristik dan perilaku tertentu.

2.2.1. Class dan Object

Class merupakan cetak biru (*blueprint*) yang mendefinisikan atribut dan metode yang dimiliki oleh suatu objek. *Object* adalah *instansiasi* dari *class* yang merepresentasikan bentuk nyata dari rancangan tersebut. Melalui konsep *class* dan *object*, pengembang dapat memodelkan berbagai entitas dalam sistem secara terstruktur dan sistematis. Dalam pengembangan *game*, setiap karakter, item, atau elemen permainan dapat direpresentasikan sebagai objek yang berasal dari *class* tertentu. Hal ini memudahkan pengelolaan data dan perilaku dalam sistem permainan.

2.2.2. Encapsulation

Encapsulation adalah konsep pengikapsulan data dan metode dalam satu kesatuan objek serta pembatasan akses langsung terhadap data tersebut. Data dalam sebuah objek hanya dapat diakses melalui metode tertentu yang telah ditentukan. Konsep ini bertujuan untuk menjaga keamanan data dan mencegah perubahan data secara tidak terkontrol. Dengan *encapsulation*, struktur program menjadi lebih terorganisir dan mudah dipelihara. Dalam *game* edukatif, *encapsulation* memungkinkan pengelolaan status objek, seperti kondisi atau nilai tertentu, secara terkontrol melalui metode yang sesuai.

2.2.3. Inheritance

Inheritance merupakan konsep pewarisan atribut dan metode dari sebuah *class* induk (*parent class*) ke *class* turunan (*child class*). Dengan *inheritance*, *class* turunan dapat menggunakan kembali atribut dan metode yang dimiliki *class* induk tanpa harus mendefinisikannya ulang. Konsep inheritance membantu mengurangi duplikasi kode dan membentuk hierarki *class* yang jelas. Dalam *game*, *inheritance* memungkinkan pengelompokan objek yang memiliki karakteristik serupa sehingga pengembangan sistem menjadi lebih efisien.

2.2.4. Polymorphism

Polymorphism adalah kemampuan suatu metode untuk memiliki bentuk atau perilaku yang berbeda pada objek yang berbeda. Dengan *polymorphism*, sebuah metode dengan nama yang sama dapat diimplementasikan secara berbeda sesuai dengan kebutuhan masing-masing objek. Konsep ini memberikan fleksibilitas dalam pengembangan sistem dan memungkinkan program berjalan secara dinamis. Dalam *game* edukatif, *polymorphism* dapat digunakan untuk menampilkan perilaku objek yang berbeda berdasarkan kondisi tertentu.

2.2.5. Abstraction

Abstraction merupakan konsep penyederhanaan sistem dengan menampilkan fitur-fitur yang relevan dan menyembunyikan detail implementasi yang tidak diperlukan oleh pengguna. *Abstraction* membantu pengembang fokus pada fungsi utama sistem tanpa harus memikirkan detail teknis yang kompleks. Dengan *abstraction*, sistem menjadi lebih mudah dipahami dan dikembangkan. Konsep ini sangat penting dalam pengembangan aplikasi berskala besar maupun kecil, termasuk *game* edukatif, karena membantu menjaga kompleksitas sistem tetap terkendali.

BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1. Deskripsi Umum Aplikasi

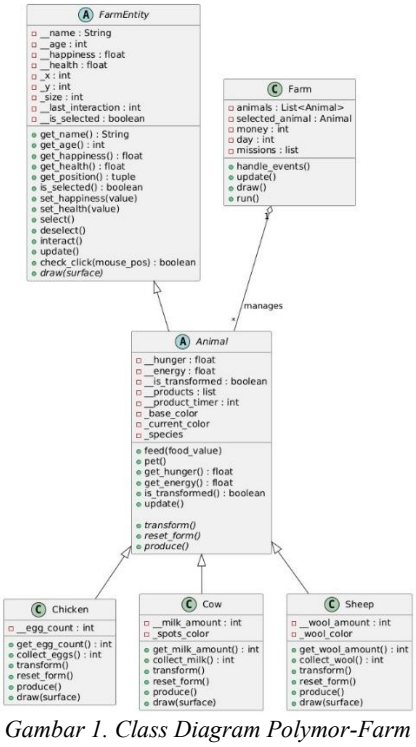
Polymor-Farm merupakan aplikasi *game* edukatif sederhana yang dirancang untuk anak sekolah dasar. *Game* ini menggunakan konsep simulasi peternakan, di mana pemain berperan sebagai pengelola peternakan yang bertugas merawat hewan ternak seperti ayam, sapi, dan domba. Aktivitas utama dalam *game* meliputi memberi makan hewan, menjaga tingkat kebahagiaan, serta mengelola hasil produksi.

Aplikasi ini dikembangkan menggunakan bahasa pemrograman *Python* dengan *library Pygame*. Polymor-Farm dirancang sebagai media pembelajaran berbasis pengalaman yang mengajarkan nilai tanggung jawab, hubungan sebab akibat, serta kemampuan berhitung sederhana melalui mekanisme permainan yang interaktif dan mudah dipahami oleh anak sekolah dasar.

3.2. Diagram Class

Class utama dalam aplikasi ini adalah *class* FarmEntity yang berfungsi sebagai *class* induk dasar (*base class*). *Class* ini menyimpan atribut dan metode umum yang dimiliki oleh seluruh entitas di dalam permainan, seperti nama, posisi, kesehatan, kebahagiaan, serta metode interaksi dan pembaruan status. Selanjutnya, *class* Animal merupakan class turunan dari FarmEntity yang menambahkan karakteristik khusus hewan, seperti tingkat kelaparan, energi, serta kemampuan transformasi dan produksi.

Class Animal memiliki beberapa *class* turunan, yaitu Chicken, Cow, dan Sheep, yang masing-masing mengimplementasikan perilaku spesifik melalui metode transform(), produce(), dan draw(). Hubungan antar *class* ini menunjukkan penerapan konsep *inheritance* dan *polymorphism*, sehingga sistem menjadi lebih terstruktur, fleksibel, dan mudah dikembangkan.



Gambar 1. Class Diagram Polymor-Farm

3.3. Implementasi Konsep OOP

Implementasi konsep Object-Oriented Programming (OOP) pada game Polymor-Farm direalisasikan langsung dalam bentuk kode program Python. Setiap konsep OOP tidak hanya dijelaskan secara teori, tetapi juga diterapkan pada objek-objek yang terdapat di dalam *game*, khususnya pada hewan ternak sebagai objek utama permainan.

3.3.1. Class dan Object

Class digunakan sebagai cetak biru (*blueprint*) untuk membentuk objek dalam game. Pada Polymor-Farm, **class FarmEntity** berperan sebagai *class* dasar yang menyimpan atribut dan perilaku umum semua entitas di farm, seperti posisi, kesehatan, dan interaksi. **Class Animal** merupakan turunan dari **FarmEntity** yang merepresentasikan hewan ternak.

```
567  class Farm:
568      """Main game class dengan composition"""
569  >   def __init__(self): ···
590
591  >   def _create_missions(self): ···
600
601  >   def add_message(self, text, color=YELLOW): ···
604
605  >   def handle_events(self): ···
627
628  >   def _handle_click(self, pos): ···
649
650  >   def _feed_selected(self): ···
658
659  >   def _pet_selected(self): ···
664
665  >   def _collect_products(self): ···
699
700  >   def _check_shop_click(self, pos): ···
724
725  >   def _check_missions(self): ···
757
758  >   def update(self): ···
796
797  >   def draw(self): ···
```

Gambar 2. Code Class Farm

Objek hewan seperti ayam, sapi, dan domba dibuat dari *class* turunan Animal, yaitu *Chicken, Cow*, dan *Sheep*, yang digunakan langsung dalam permainan sebagai objek interaktif.

```
120  class Animal(FarmEntity):
121  >   """ ···
125  >   def __init__(self, name, x, y, base_color, species):
138
139  >   def get_hunger(self): ···
141
142  >   def get_species(self): ···
144
145  >   def is_transformed(self): ···
147
148  >   def get_products(self): ···
151
152  >   def get_energy(self): ···
154
155  >   def feed(self, food_value): ···
166
167  >   def pet(self): ···
171
172  >   def _check_transformation(self): ···
```

Gambar 3. Code Class Animal

3.3.2. Encapsulation

Encapsulation diterapkan dengan membatasi akses langsung terhadap atribut menggunakan *modifier private* (__). Atribut seperti kebahagiaan, kesehatan, dan kelaparan tidak dapat diubah secara langsung, melainkan melalui *method* tertentu (*getter dan setter*).

```
    # Getter methods - cara aman akses private data (ENCAPSULATION)
    def get_name(self):
        return self.__name

    def get_age(self):
        return self.__age

    def get_happiness(self):
        return self.__happiness

    def get_health(self):
        return self.__health

    def get_position(self):
        return (self._x, self._y)

    def is_selected(self):
        return self.__is_selected

    # Setter methods dengan VALIDASI (ENCAPSULATION)
    def set_happiness(self, value):
        """Setter dengan validasi - happiness harus 0-100"""
        self.__happiness = max(0, min(100, value))

    def set_health(self, value):
        """Setter dengan validasi - health harus 0-100"""
        self.__health = max(0, min(100, value))
```

Gambar 4. Code Bagian Encapsulation

Penerapan *encapsulation* ini bertujuan untuk menjaga konsistensi data dan memastikan perubahan kondisi hewan hanya terjadi melalui mekanisme yang telah ditentukan dalam sistem.

3.3.3. Inheritance

Inheritance diterapkan dengan menjadikan **class Animal** sebagai *class* induk bagi **class Chicken, Cow, dan Sheep**. *Class* turunan tersebut mewarisi atribut dan *method* dari *class* induk tanpa perlu mendefinisikan ulang.

```
class Chicken(Animal):
    """
    Chicken class - inherit dari Animal (INHERITANCE).
    Implementasi POLYMORPHISM lewat transform() yang berbeda.
    """
    def __init__(self, x, y):
        super().__init__("Ayam", x, y, WHITE, "chicken")
        self._size = 50
        self.__egg_count = 0  # Private counter

    def get_egg_count(self):
        return self.__egg_count

    def transform(self):
```

Gambar 5. Code Bagian Inheritance

Melalui *inheritance*, setiap jenis hewan memiliki karakteristik dasar yang sama, seperti rasa lapar, kebahagiaan, dan kemampuan berinteraksi, namun tetap dapat memiliki perilaku khusus sesuai jenisnya.

3.3.4. Polymorphism

Polymorphism diterapkan melalui *method* yang sama tetapi memiliki implementasi berbeda pada setiap *class* turunan. Pada *game* Polymor-Farm, *method* seperti produce(), transform(), dan draw() *dioverride* oleh masing-masing *class* hewan.

```
class Chicken(Animal):
    def transform(self):
        """
        POLYMORPHISM! Chicken transform jadi GOLDEN CHICKEN.
        Override method dari parent dengan behavior berbeda.
        """
        self._current_color = GOLD
        self._size = 60

    def reset_form(self):
        """Reset ke bentuk normal"""
        self._current_color = self._base_color
        self._size = 50

    def produce(self):
        """
        POLYMORPHISM! Chicken produce telur.
        Setiap animal produce berbeda.
        """
        if self.get_hunger() > 50 and self.get_energy() > 30:
            if self.is_transformed():
                self.__egg_count += 2  # Golden chicken produce lebih banyak!
            else:
                self.__egg_count += 1
```

Gambar 6. Code Bagian Polymorphism

Sebagai contoh, ayam menghasilkan telur, sapi menghasilkan susu, dan domba menghasilkan wool. Selain itu, setiap hewan memiliki bentuk transformasi dan tampilan visual yang berbeda meskipun menggunakan *method* dengan nama

yang sama.

3.3.5. Abstraction

Abstraction diterapkan dengan mendefinisikan *method* abstrak pada **class FarmEntity** dan **Animal** menggunakan **decorator @abstractmethod**. *Method* seperti draw(), produce(), dan transform() hanya dideklarasikan pada *class* induk tanpa implementasi detail.

Gambar 7. Code Bagian Abstraction

Implementasi *method* tersebut kemudian diwajibkan pada *class* turunan, sehingga detail teknis disembunyikan dan hanya fungsi utama yang ditentukan pada level abstrak.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1. Fitur Utama

Game Polymor-Farm memiliki beberapa fitur utama yang dirancang untuk mendukung pembelajaran interaktif bagi anak sekolah dasar:

1. **Sistem Perawatan Hewan:**
   Pemain dapat berinteraksi langsung dengan hewan ternak (ayam, sapi, domba) dengan cara memberi makan untuk mengurangi tingkat kelaparan. Setiap hewan memiliki indikator visual yang menunjukkan status kesehatan, kebahagiaan, dan kelaparan.
2. **Sistem Produksi:**
   Setelah hewan mencapai tingkat transformasi tertentu, hewan dapat menghasilkan produk. Ayam menghasilkan telur, sapi menghasilkan susu, dan domba menghasilkan wool. Produk yang dihasilkan dapat dikumpulkan dan dihitung oleh pemain.
3. **Manajemen Sumber Daya:**
   Pemain harus mengelola persediaan makanan dan memutuskan kapan harus memberi makan setiap hewan. Hal ini melatih kemampuan berhitung sederhana dan pengambilan keputusan.
4. **Indikator Status Real-time:**
   Setiap hewan memiliki indikator yang menampilkan tingkat kesehatan, kebahagiaan, dan kelaparan secara visual, sehingga anak dapat memahami kondisi hewan dengan mudah..

4.2. Cara Penggunaan

Berikut adalah langkah-langkah penggunaan game Polymor-Farm:

1. Memulai Game:
   Jalankan aplikasi dengan mengeksekusi file Python utama. Layar awal akan menampilkan farm dengan beberapa hewan ternak yang siap untuk dirawat.
2. Memilih Hewan:
   Klik pada hewan yang ingin dirawat. Hewan yang dipilih akan ditandai dengan highlight atau border khusus.
3. Memberi Makan:
   Setelah memilih hewan, tekan tombol keyboard yang telah ditentukan (tombol F) untuk memberi makan hewan. Tingkat kelaparan hewan akan berkurang dan kebahagiaan akan meningkat ketika di tekan tombol keyboard yang telah ditentukan (tombol P).
4. Memantau Status:
   Perhatikan kotak indikator di atas setiap hewan untuk memantau kondisi mereka. kotak merah menunjukkan kesehatan, kotak merah muda menunjukkan kebahagiaan dan kotak hijau menunjukkan tingkat kenyang.
5. Mengumpulkan Produk:
   Ketika hewan telah berkembang dan siap berproduksi, tekan tombol keyboard yang telah ditentukan (tombol P).untuk mengumpulkan telur, susu, atau wool.
6. Manajemen Farm:
   Kelola waktu dan sumber daya dengan baik agar semua hewan tetap sehat dan bahagia. Hewan yang tidak dirawat akan mengalami penurunan status dan tidak

dapat berproduksi secara optimal.

4.3. Tampilan

Game Polymor-Farm dirancang dengan antarmuka sederhana dan ramah anak:

1. Layar Utama (Main Screen):
   Menampilkan area farm dengan latar belakang padang rumput hijau. Terdapat 3-5 hewan ternak yang tersebar di area farm dengan posisi yang dapat dikustomisasi.
2. Panel Kontrol:
   Terletak di bagian bawah atau samping layar, berisi tombol-tombol interaksi seperti:
   - Tombol keyboard "F" untuk memberi makan
   - Tombol keyboard "C" untuk mengumpulkan produk
   - Tombol keyboard "P" untuk memberi kebahagiaan
3. Status Bar Hewan:
   Setiap hewan memiliki 3 bar horizontal di atas tubuhnya:
   - Bar merah: Kesehatan (Health)
   - Bar merah muda: Kebahagiaan (Happiness)
   - Bar hijau: Tingkat kenyang (Fullness)
4. Sprite Hewan:
   Tampilan visual hewan menggunakan sprite sederhana dengan warna cerah dan bentuk yang mudah dikenali:
   - Ayam: Sprite berwarna putih/kuning dengan transformasi warna
   - Sapi: Sprite berwarna coklat/merah muda dengan transformasi warna
   - Domba: Sprite berwarna putih/merah muda dengan transformasi warna
5. Panel Informasi:
   Di pojok layar terdapat panel yang menampilkan:
   - Jumlah produk yang telah dikumpulkan
   - Skor atau poin pemain
   - Waktu bermain (berapa hari yang telah berlalu)
   - Jumlah uang yang dimiliki
6. Animasi Visual:
   Terdapat animasi sederhana seperti:
   - Hewan bergerak
   - Efek partikel saat memberi makan
   - Transisi bertahap saat transformasi hewan
   - Ikon produk muncul di atas hewan saat siap dipanen

Desain visual menggunakan warna-warna cerah dan elemen grafis yang sederhana agar mudah dipahami oleh anak sekolah dasar. Semua elemen UI menggunakan ikon dan simbol yang intuitif untuk meminimalkan penggunaan teks yang kompleks..

BAB V
PENUTUP

5.1. Kesimpulan

Game Polymor-Farm berperan sebagai media pembelajaran interaktif yang efektif bagi anak sekolah dasar, mengajarkan tanggung jawab, konsep sebab-akibat, dan kemampuan berhitung sederhana melalui simulasi perawatan hewan ternak. Penerapan prinsip Object-Oriented Programming (OOP) seperti *class*, *object*, *encapsulation, inheritance, polymorphism,* dan *abstraction* diterapkan secara konsisten, menghasilkan struktur kode yang terorganisir dan fitur game yang mendukung pembelajaran, seperti sistem perawatan hewan, transformasi visual, produksi sumber daya, dan indikator status *real-time*. Kombinasi ini menciptakan pengalaman belajar yang menyenangkan sekaligus bermakna.

Selain sebagai hiburan, Polymor-Farm telah menunjukkan potensi sebagai sarana pembelajaran yang edukatif dan berbasis pengalaman. Fitur-fitur yang ada bekerja secara sinergis untuk membimbing anak memahami proses dan konsekuensi tindakan dalam konteks yang mudah dipahami, sehingga *game* ini tidak hanya menghibur tetapi juga memperkuat kemampuan kognitif dan afektif pemain.

5.2. Saran

Pengembangan Polymor-Farm selanjutnya dapat menambah variasi hewan dan kondisi *farm*, serta mengintegrasikan konten edukatif yang lebih eksplisit, seperti kuis atau *mini-game* berhitung, untuk memperluas pengalaman belajar. Penambahan mode *multiplayer* atau fitur sosial juga dapat mengajarkan kerja sama dan komunikasi antar pemain, sementara sistem *reward* dan pencapaian meningkatkan motivasi anak untuk menyelesaikan tugas. Selain itu, pengujian dengan pengguna nyata dan optimasi kode sangat penting untuk memastikan *gameplay* berjalan lancar dan sesuai dengan kebutuhan anak sekolah dasar. Penambahan opsi aksesibilitas, dokumentasi kode, dan pengembangan narasi atau tutorial interaktif juga akan memperkuat pengalaman belajar, sehingga Polymor-Farm menjadi media pembelajaran yang lebih komprehensif, menarik, dan inklusif.

DAFTAR PUSTAKA

HelloSehat. (2024). 8 Pilihan Permainan untuk Anak SD yang Bermanfaat. Diakses dari https://hellosehat.com/parenting/anak-6-sampai-9-tahun/perkembangan-anak/game-edukasi-anak-sd/

LearningRoom.id. (n.d.). Permainan Edukasi yang Bantu Anak SD dan SMP Belajar. Diakses dari https://learningroom.id/media/blog/detail/permainan-edukasi-anak-sd-belajar

Mitrapost.com. (2025). 4 Rekomendasi Game Edukasi untuk Anak Sekolah Dasar. Diakses dari https://mitrapost.com/2025/12/04/4-rekomendasi-game-edukasi-untuk-anak-sekolah-dasar/

Telkomsel. (2025). Game Edukasi Terbaik 2025: Serunya Belajar Sambil Bermain! Diakses dari https://www.telkomsel.com/jelajah/jelajah-lifestyle/game-edukasi-terbaik-2025-serunya-belajar-sambil-bermain

RevoU. (n.d.). Apa itu OOP? Arti, Fungsi, Contoh. Diakses dari https://www.revou.co/kosakata/oop

Kumar, M. (2025). Mastering Object-Oriented Programming (OOP) in Java: Encapsulation, Inheritance, Polymorphism, and Abstraction. Medium. Diakses dari https://medium.com/@manishkumar_75473/mastering-object-oriented-programming-oop-in-java

Susanto, G. A. (2020). Belajar Konsep OOP (Object Oriented Programming): Inheritance dan Contoh Penerapannya dalam Bahasa Kotlin. Medium. Diakses dari https://galangaji.medium.com/2-belajar-konsep-oop-object-oriented-programming-inheritance

Unesa. (n.d.). Object-Based Programming: Definition, Concept and Types of Programming Languages. Diakses dari https://si.ft.unesa.ac.id/post/pemrograman-berbasis-objek-pengertian-konsep-dan-macam-macam-bahasa-pemrogramannya

Pygame.org. (n.d.). Pygame Front Page — pygame v2.6.0 documentation. Diakses dari https://www.pygame.org/docs/

PGSD BINUS. (2021). Implementasi Teori Belajar Kognitivisme dalam Pandangan Jean Piaget dan Jerome Bruner. Diakses dari https://pgsd.binus.ac.id/2021/07/08/implementasi-teori-belajar-kognitivisme-dalam-pandangan-jean-piaget-dan-jerome-bruner/

Pendas. (2022). Implikasi Teori Perkembangan Kognitif Piaget pada Anak Usia Sekolah Dasar dalam Pembelajaran Bahasa Indonesia. Jurnal Ilmiah Pendidikan Dasar. Diakses dari https://journal.unpas.ac.id/index.php/pendas/article/view/6564

Halodoc. (2024). 4 Tahap Perkembangan Kognitif Anak Sesuai Teori Piaget. Diakses dari https://www.halodoc.com/artikel/4-tahap-perkembangan-kognitif-anak-sesuai-teori-piaget

Hibatullah, M. F. (2021). Penerapan Konsep Object Oriented Programming dalam Perancangan Sistem Informasi. Jurnal Teknologi Informasi.