**LAPORAN PROJECT *MAZE RUNNER ARCADE***

Disusun untuk memenuhi tugas UAS Mata Kuliah Pemograman Berbasis Objek

**Mata Kuliah:**

Pemograman Berbasis Objek

**Dosen Pengampu:**

I Gde Agung Sri Sidhimantra, S.Kom., M.Kom.

**Disusun Oleh:**

1. Mada Permata 23091397179
2. Zaidan Mudzaky Juan Kusuma 23091397194
3. Ardli Kafi Murobby 23091397196



**PROGRAM STUDI MANAJEMEN INFORMATIKA**

**FAKULTAS VOKASI**

**UNIVERSITAS NEGERI SURABAYA**

**2024**

**Daftar Isi**

[**1.** **Pendahuluan** 3](#_Toc186539022)

[**1.1.** **Latar Belakang** 3](#_Toc186539023)

[**1.2.** **Tujuan Proyek** 4](#_Toc186539024)

[**1.3.** **Ruang Lingkup** 4](#_Toc186539025)

[**2.** **Penerapan Konsep Pemrograman Berorientasi Objek (OOP)** 5](#_Toc186539026)

[**2.1.** **Abstraksi** 5](#_Toc186539027)

[**2.2.** **Enkapsulasi** 5](#_Toc186539028)

[**2.3.** **Pewarisan (Inheritance)** 5](#_Toc186539029)

[**2.4.** **Polimorfisme** 5](#_Toc186539030)

[**2.5.** **Komposisi** 6](#_Toc186539031)

[**3.** **Class Diagram** 7](#_Toc186539032)

[**4.** **Alur Aplikasi (Game Loop)** 9](#_Toc186539033)

[***5.*** **Tantangan dan Pembelajaran Selama Pengembangan Game *Maze Runner Arcade*** 11](#_Toc186539034)

[**5.1.** **Tantangan Teknis** 11](#_Toc186539035)

[**5.3.** **Tantangan Non-Teknis** 11](#_Toc186539036)

[**5.4.** **Solusi yang Diterapkan** 12](#_Toc186539037)

[**5.5.** **Pembelajaran dari Proyek *Maze Runner Arcade*** 13](#_Toc186539038)

[**6.** **Kesimpulan** 14](#_Toc186539039)

[**7.** **Lampiran** 16](#_Toc186539040)

[**7.1.** **Lampiran 1 : Source Code Proyek** 16](#_Toc186539041)

[***7.2.*** **Lampiran 2 : Tampilan Game *Maze Runner Arcade*** 26](#_Toc186539042)

# **Pendahuluan**

## **Latar Belakang**

Proyek *'Maze Runner Arcade'* dikembangkan sebagai implementasi praktis dari konsep Pemrograman Berorientasi Objek (OOP). Dalam era digital, game tidak lagi dipandang hanya sebagai hiburan semata, namun juga sebagai media pembelajaran yang interaktif dan efektif. Pengembangan game, khususnya, melatih kemampuan pemecahan masalah (problem-solving), pemikiran logis, dan kreativitas. Proyek ini tidak hanya tentang menciptakan game, tetapi juga tentang belajar bagaimana mengaplikasikan konsep-konsep pemrograman dalam konteks yang menarik dan menantang.

Pemrograman Berorientasi Objek (OOP) merupakan paradigma pemrograman yang sangat relevan dan penting dalam pengembangan perangkat lunak modern. OOP membantu dalam mengorganisir kodingan, meningkatkan reusability, dan maintainability. Sebagai mahasiswa D4 Manajemen Informatika, pemahaman yang mendalam tentang OOP sangat krusial untuk mengembangkan aplikasi yang terstruktur, fleksibel, dan mudah dipelihara. Proyek ini memberikan kesempatan bagi kami untuk mempraktikkan konsep-konsep OOP secara nyata dalam sebuah aplikasi.

Dalam konteks tugas mata kuliah Pemrograman Berbasis Objek, penggunaan Python sebagai bahasa pemrograman adalah sebuah ketentuan wajib. Python dipilih karena sintaksnya yang mudah dipahami dan dibaca, membuatnya ideal untuk mempelajari konsep-konsep OOP. Selain itu, Python juga memiliki berbagai pustaka dan framework pendukung yang sangat berguna dalam berbagai aspek pengembangan aplikasi, termasuk game development. Pustaka Pygame, yang kami gunakan dalam proyek ini, dipilih karena open-source, kemudahan dalam mengimplementasikan grafis, input, dan game loop, serta dokumentasinya yang lengkap. Kombinasi Python dan Pygame sangat ideal untuk proyek game 2D sederhana seperti *'Maze Runner Arcade'*, sekaligus memenuhi kriteria tugas yang mengharuskan penggunaan Python.

Proyek *'Maze Runner Arcade'* ini dikembangkan sebagai salah satu bentuk evaluasi pemahaman kami terhadap konsep-konsep OOP dalam mata kuliah Pemrograman Berbasis Objek. Melalui proyek ini, kami diharapkan dapat menunjukkan kemampuan kami dalam merancang, mengimplementasikan, dan mengaplikasikan konsep OOP dalam suatu aplikasi nyata. Proyek ini menjadi output yang konkret untuk mengukur pemahaman kami mengenai OOP dan relevansinya dengan tugas UAS yang telah diberikan, serta bagaimana kami menggunakan Python sebagai bahasa pemrograman utama dalam implementasinya.

## **Tujuan Proyek**

Tujuan utama dari proyek ini adalah untuk mengaplikasikan konsep-konsep Pemrograman Berbasis Objek (OOP) dalam pengembangan game sederhana.

Tujuan spesifiknya meliputi:

1) Merancang dan mengimplementasikan struktur kelas yang mencerminkan konsep OOP seperti abstraksi, enkapsulasi, pewarisan, dan polimorfisme.

2) Membuat game interaktif yang melibatkan pergerakan karakter, interaksi dengan objek lain (bintang), dan penentuan kondisi menang/kalah.

3) Mempelajari dan memanfaatkan pustaka Pygame untuk mengimplementasikan elemen grafis dan interaksi game.

4) Meningkatkan kemampuan kami dalam memahami dan menerapkan prinsip-prinsip OOP dalam proyek nyata.

5) Menyelesaikan tugas Ujian Akhir Semester (UAS) mata kuliah Pemrograman Berbasis Objek.

## **Ruang Lingkup**

Proyek game *“Maze Runner Arcade”* memiliki ruang lingkup yang mencakup beberapa aspek utama:

1) Implementasi konsep OOP seperti abstraksi, enkapsulasi, pewarisan, polimorfisme, dan komposisi.

2) Pembuatan labirin sederhana dengan dinding dan ruang kosong menggunakan struktur data array.

3) Implementasi pergerakan pemain menggunakan input keyboard.

4) Pembuatan NPC (Non-Player Character) penjaga yang bergerak mengikuti pemain dengan algoritma BFS.

5) Pengumpulan bintang sebagai target dalam permainan.

6) Pembuatan menu utama, menu kesulitan, dan menu menang/kalah.

7) Implementasi grafik sederhana menggunakan Pygame.

Proyek yang kami buat ini tidak mencakup fitur-fitur yang kompleks seperti editor level, multiplayer, atau sistem scoring yang rumit.

# **Penerapan Konsep Pemrograman Berorientasi Objek (OOP)**

## **Abstraksi**

Konsep abstraksi diimplementasikan melalui penggunaan kelas abstrak *GameEntity*. Kelas ini mendefinisikan interface umum, berupa metode abstrak *draw()* dan *update()*, yang harus diimplementasikan oleh semua kelas turunan yang merepresentasikan entitas dalam game seperti *Player*, *Guard, Maze*, dan *Button*. Dengan menggunakan abstraksi, kami dapat menyembunyikan detail implementasi dari setiap entitas dan hanya berfokus pada interface yang dibutuhkan untuk berinteraksi dengan mereka.

## **Enkapsulasi**

Konsep enkapsulasi diterapkan dengan mengemas atribut (data) dan metode (perilaku) yang terkait ke dalam setiap kelas. Contohnya, kelas *Player* memiliki atribut posisi *(\_x, \_y)*, kecepatan *(\_speed),* dan animasi, serta metode untuk mengatur pergerakan, tampilan, dan update animasi. Atribut-atribut internal kelas hanya dapat diakses dan diubah melalui metode-metode yang telah didefinisikan dalam kelas tersebut, sehingga data terlindungi dari akses dan modifikasi yang tidak diinginkan dari luar kelas.

## **Pewarisan (Inheritance)**

Konsep pewarisan diterapkan dengan membuat kelas *Player*dan *Guard*sebagai turunan dari kelas abstrak *Character*. Kelas Character menyediakan atribut dan metode dasar seperti posisi, kecepatan, dan metode *draw()* dan *update()*, yang kemudian diwarisi oleh kelas *Player* dan *Guard*. Setiap kelas turunan dapat menambahkan dan menyesuaikan implementasi metode yang diwarisi sesuai dengan kebutuhan masing-masing karakter. Hal ini memungkinkan penggunaan kembali kodingan dan memfasilitasi pembentukan hierarki kelas yang jelas.

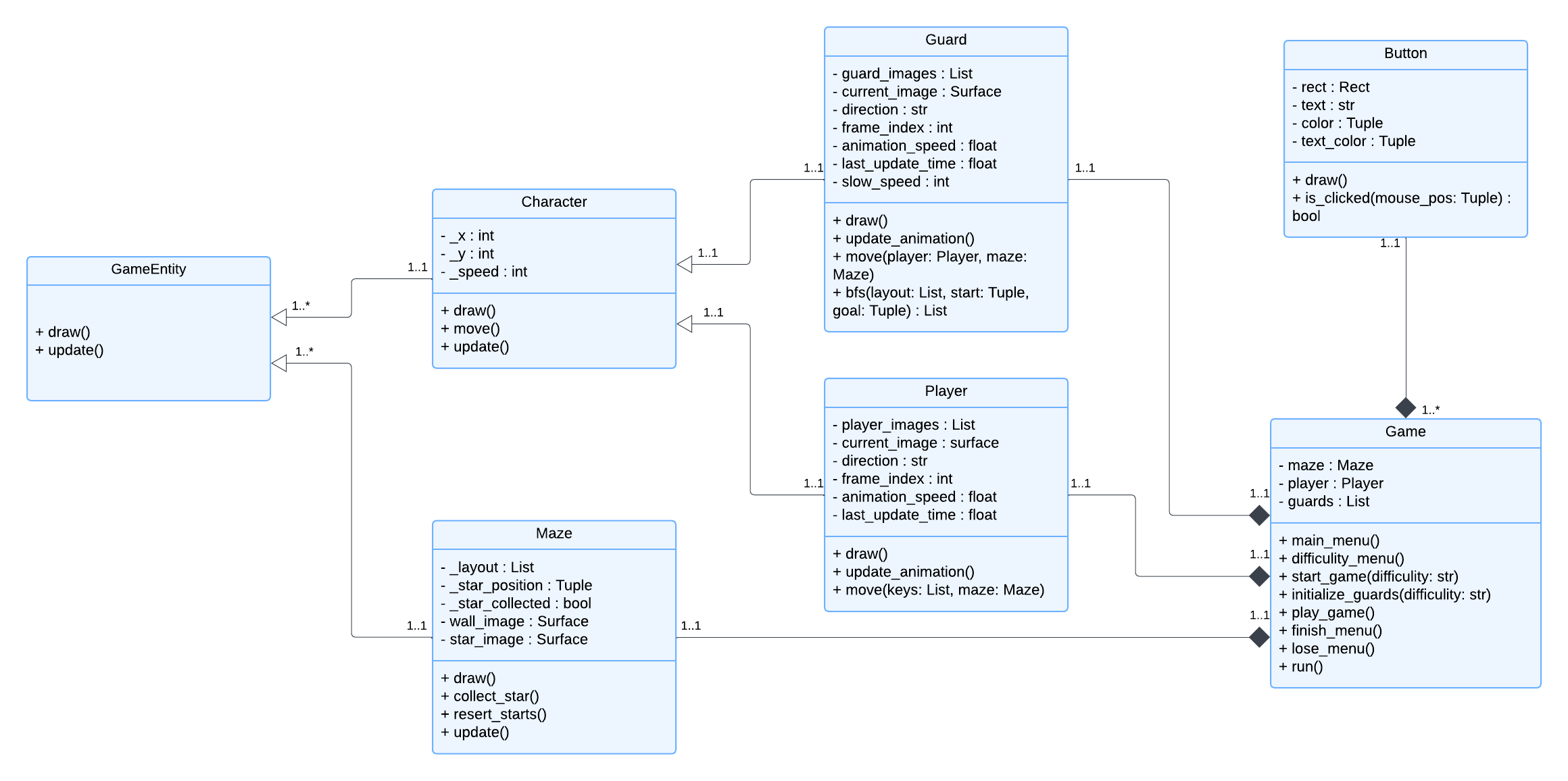
## **Polimorfisme**

Konsep polimorfisme diterapkan melalui penggunaan metode *draw()*. Metode ini diimplementasikan secara berbeda oleh setiap kelas turunan dari *GameEntity*. Misalnya, kelas *Maze* mengimplementasikan metode *draw()* untuk menggambar labirin, kelas *Player* untuk menggambar karakter pemain, dan kelas *Guard* untuk menggambar karakter penjaga. Dengan menggunakan polimorfisme, kita dapat memanggil metode *draw()* pada semua objek secara seragam, tanpa mempedulikan jenis objeknya. Implementasi spesifik dari *draw()* pada setiap objek akan ditangani oleh kelas masing-masing.

## **Komposisi**

Konsep komposisi diterapkan dalam kelas Game, yang memiliki atribut berupa objek *Maze*, objek *Player*, dan daftar objek *Guard*. Kelas Game bertanggung jawab untuk mengelola dan mengatur interaksi antara objek-objek ini. Kelas Game menggunakan objek-objek lain dari kelas yang berbeda untuk membangun suatu entitas game yang lebih kompleks. Dengan komposisi, kita dapat membangun aplikasi yang terdiri dari bagian-bagian yang lebih kecil dan terstruktur.

# **Class Diagram**



**Gambar 3.1** Class Diagram Game Maze Runner Arcade

Pada bagian ini, kami menyajikan diagram kelas UML (Gambar 3.1) yang menggambarkan struktur kelas-kelas utama dalam game 'Maze Runner Arcade'. Diagram kelas ini memvisualisasikan bagaimana kelas-kelas diorganisasikan, atribut dan metode yang mereka miliki, serta hubungan antar kelas tersebut. Diagram kelas ini sangat penting karena merepresentasikan desain berorientasi objek yang kami terapkan dalam proyek ini.

Secara umum, diagram kelas ini menunjukkan bahwa terdapat kelas abstrak GameEntity yang menjadi base class dari semua entitas game yang dapat digambar. Kelas Character adalah kelas abstrak yang merepresentasikan semua karakter, dan kelas Player dan Guard mewarisi sifat dari kelas Character, dengan penambahan behavior yang lebih spesifik. Terdapat juga kelas Maze, yang merepresentasikan labirin, dan kelas Button yang merepresentasikan tombol interaktif. Kelas Game adalah kelas yang mengontrol seluruh alur game.

Kelas-kelas utama yang terdapat dalam diagram kelas ini adalah:

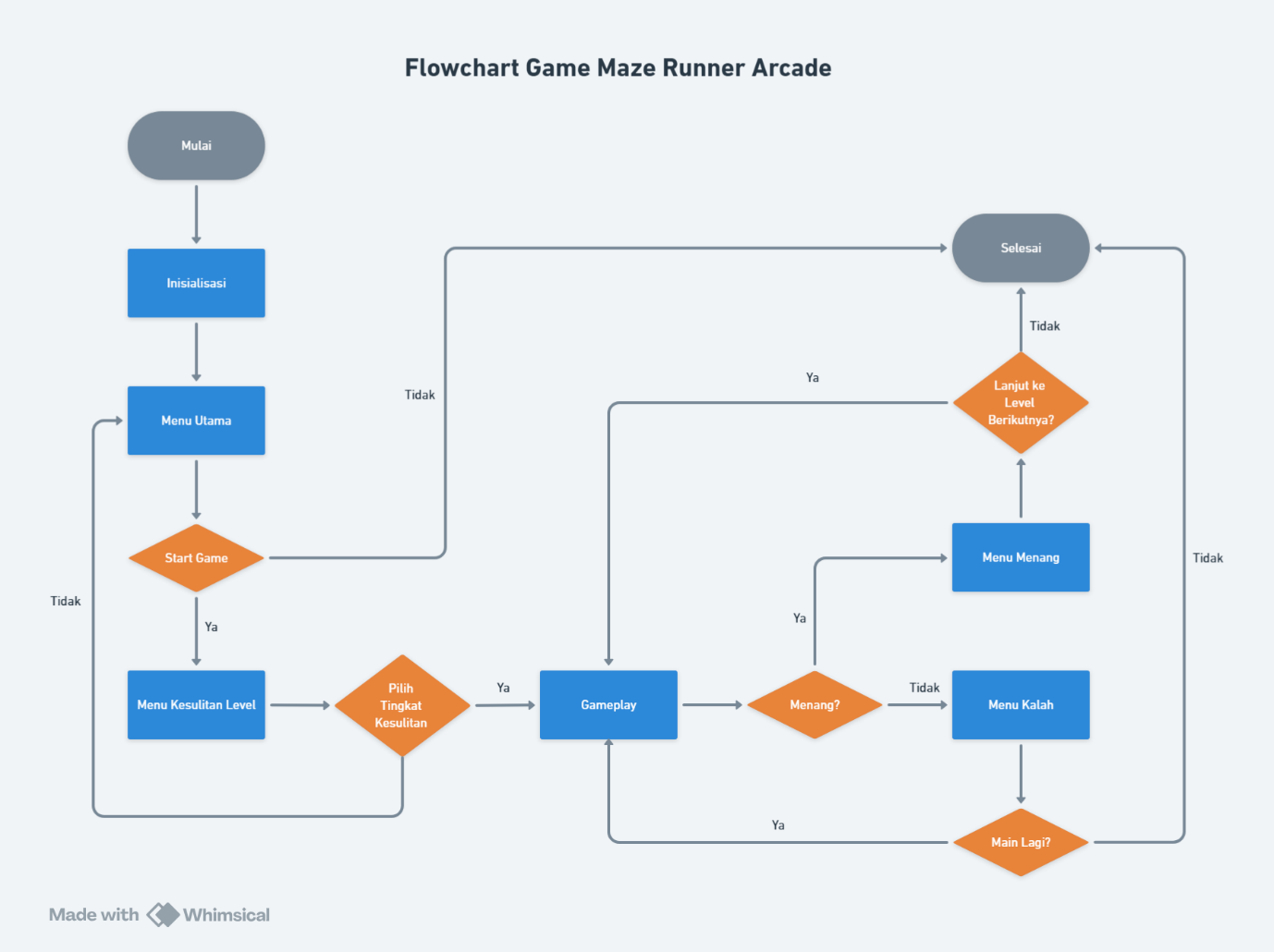
* **GameEntity**: Kelas abstrak yang menjadi *base class* dari semua entitas *game*, dengan metode *draw()* dan *update().*
* **Character**: Kelas abstrak untuk semua karakter dalam *game*, dengan atribut posisi *(\_x, \_y)* dan kecepatan *(\_speed)* serta metode *draw(), update(),* dan*move().*
* **Player**: Kelas yang mewarisi dari kelas Character untuk merepresentasikan pemain, dengan atribut gambar animasi dan metode pergerakan menggunakan *keyboard*.
* **Guard**: Kelas yang mewarisi dari kelas Character untuk merepresentasikan penjaga, dengan atribut gambar animasi dan metode pergerakan berdasarkan algoritma BFS.
* **Maze**: Kelas untuk merepresentasikan labirin, dengan atribut layout, star\_positions, dan gambar *wall\_image*, serta metode *draw()*, *collect\_star()* dan *reset\_stars().*
* **Button**: Kelas untuk merepresentasikan tombol interaktif dengan atribut rect, text, color, text\_color, dan metode *draw()* dan *is\_clicked().*
* **Game**: Kelas yang mengelola seluruh *game*, yang memiliki objek *Maze*, *Player*, dan daftar *Guard*, serta metode-metode untuk *menu*, *gameplay*, dan kondisi akhir.

Relasi antar kelas dalam diagram ini meliputi:

* **Pewarisan (Inheritance):** Kelas *Player* dan *Guard* mewarisi dari kelas abstrak *Character*, dan kelas *Character* mewarisi dari kelas abstrak *GameEntity*.
* **Komposisi (Composition):** Kelas Game memiliki objek *Maze*, *Player*, dan daftar *Guard* sebagai bagian dari dirinya.
* **Asosiasi:** Kelas *Game* berasosiasi dengan *Button*.

Diagram kelas ini memberikan visualisasi yang jelas tentang struktur OOP yang kami terapkan dalam proyek ini dan memberikan pemahaman yang jelas mengenai bagaimana setiap kelas berinteraksi dan berkontribusi dalam sistem secara keseluruhan.

# **Alur Aplikasi (Game Loop)**



**Gambar 4.1** Flowchart Game Maze Runner Arcade

Gambar di atas adalah flowchart dari game yang telah kami buat, game dapat di mulai ketika kita menjalankan program atau kodingannya, ketika kodingan dijalankan, kodingan akan melakukan inisialisasi yang mana akan menunjukkan proses inisialisasi Pygame, *window*, *font*, dan pembuatan objek game.

Kemudian, kita sudah berhasil di inisialisasi akan muncul antarmuka game nya yaitu menu utama yang terdapat dua pilihan yaitu *start game* dan *quit game*, ketika pemain memilih *start game* maka akan di hadapkan dengan menu untuk memilih tingkat kesulitan gamenya, dan ketika pemain memilih opsi quit game, maka jendela antarmuka gamenya akan tertutup.

Setelah pemain memutuskan untuk memainkan gamenya setelah memilih level yang ditentukan, maka pemain sudah bisa langsung memainkan game nya, setelah memainkan game tersebut dan jika pemain menang, maka pemain akan di hadapkan dengan menu ketika menang dan sebaliknya. Namun, jika pemain berhasil memenangkan permainan, pemain dapat melanjutkan ke level selanjutnya, dan jika kalah pemain bisa kembali ke menu sebelumnya atau memainkan gamenya kembali.

# **Tantangan dan Pembelajaran Selama Pengembangan Game *Maze Runner Arcade***

Proses pengembangan proyek *'Maze Runner Arcade'* bukanlah tanpa tantangan, kami menghadapi berbagai tantangan, baik dari segi teknis maupun non-teknis. Adapun berikut adalah detail tantangan, solusi yang kami terapkan, dan pembelajaran yang kami dapatkan

selama proses pengembangan proyek game *’Maze Runner Arcade’* :

## **Tantangan Teknis**

Kami menghadapi beberapa tantangan teknis yang cukup signifikan selama pengembangan. Tantangan tersebut meliputi:

* + **Implementasi Algoritma BFS**: Mengimplementasikan algoritma *Breadth-First Search (BFS)* untuk pergerakan guard yang efisien dan efektif. Kami perlu memahami logika algoritma ini dengan baik dan menerapkannya dalam kodingan dengan benar.
  + **Animasi Karakter**: Membuat animasi karakter pemain dan *guard* yang terlihat halus dan responsif. Kami perlu mensinkronkan gambar animasi dengan pergerakan karakter dan memastikan perpindahan antar frame berjalan dengan lancar.
  + **Penanganan Input Keyboard**: Memastikan game merespons *input* dari keyboard dengan cepat, tepat, dan tanpa *delay*. Kami perlu memastikan setiap tombol arah dapat menggerakkan pemain sesuai arah yang ditekan.
  + **Manajemen State Game**: Mengelola transisi antar state game ( menu utama, menu kesulitan, gameplay, dan menu akhir) dengan benar dan efisien agar tidak terjadi *error*.
  + **Pengaturan Posisi Objek**: Memastikan setiap objek seperti pemain, guard, labirin, dan bintang berada pada posisi yang tepat dan tidak saling tumpang tindih. Kami perlu melakukan perhitungan yang tepat untuk mengatur posisi objek dalam *grid* labirin.

## **Tantangan Non-Teknis**

Selain tantangan teknis, kami juga menghadapi tantangan non-teknis yang juga penting dalam keberhasilan proyek ini. Tantangan tersebut adalah:

* + **Pembagian Tugas**: Memastikan pembagian tugas antar anggota kelompok merata dan sesuai dengan kemampuan masing-masing. Kami perlu memastikan setiap anggota kelompok memiliki tanggung jawab yang jelas dan terdefinisi.
  + **Koordinasi Anggota Kelompok**: Memastikan koordinasi antar anggota kelompok berjalan dengan baik selama proses coding, testing, dan debugging. Komunikasi yang aktif dan teratur sangat penting untuk menghindari konflik dan overlap tugas.
  + **Manajemen Waktu**: Mengelola waktu agar semua tugas dapat diselesaikan dalam deadline yang telah ditentukan. Kami perlu membuat jadwal kerja yang terstruktur dan memprioritaskan tugas-tugas penting.

## **Solusi yang Diterapkan**

* **Implementasi BFS**: Kami mempelajari referensi tentang algoritma BFS yang ada di internet, kemudian kami juga melakukan step-by-step debugging dan pengujian, serta melakukan iterasi pada kodingan.
* **Animasi Karakter**: Kami juga bereksperimen dengan berbagai timing animasi dan gambar yang tepat. Kami juga menggunakan logika pergerakan yang sesuai dengan arah pergerakan karakter.
* **Penanganan Input**: Dalam mengatasi permasalahan ini, kami menggunakan *pygame.key.get\_pressed()* supaya input keyboard dapat diproses secara simultan, dan menggunakan struktur *if-else* untuk memproses tombol yang ditekan.
* **Manajemen State Game**: Kami membuat fungsi khusus untuk setiap state dan transisi antar state, memastikan bahwa game berpindah dengan benar dan tanpa error.
* **Pengaturan Posisi Objek**: Kami melakukan perhitungan yang tepat dalam menentukan koordinat objek, dan memanfaatkan grid-based system pada labirin.
* **Pembagian Tugas, Koordinasi, Manajemen Waktu**: Kami melakukan komunikasi teratur, membuat meeting terjadwal, membuat jadwal kerja yang rinci, memprioritaskan tugas penting, dan membuat timeline proyek.

## **Pembelajaran dari Proyek *Maze Runner Arcade***

Melalui proyek ini, kami belajar banyak hal baik dari aspek teknis maupun non-teknis, diantaranya:

* Penerapan konsep OOP (abstraksi, enkapsulasi, pewarisan, polimorfisme, dan komposisi) sangat membantu dalam membuat struktur kodingan yang modular dan mudah di-maintain.
* Pustaka Pygame sangat berguna dalam pengembangan *game* 2D, dan kami berhasil memanfaatkan berbagai fitur yang ada didalamnya.
* Algoritma BFS sangat berguna untuk mencari jalur terpendek.
* Pengembangan game membutuhkan perencanaan yang matang, manajemen waktu yang baik, dan kerja kelompok yang solid.
* Proses debugging memerlukan kesabaran, ketelitian, dan kemampuan problem solving.
* Testing sistem adalah tahapan penting untuk memastikan bahwa aplikasi berjalan dengan benar sebelum nantinya dikumpulkan.

# **Kesimpulan**

Proyek 'Maze Runner Arcade' telah berhasil diimplementasikan sebagai aplikasi game 2D sederhana yang bertujuan untuk menerapkan konsep-konsep Pemrograman Berorientasi Objek (OOP) dalam pengembangan game. Proyek ini dikerjakan sebagai tugas akhir mata kuliah Pemrograman Berbasis Objek dan telah memberikan kami pengalaman yang berharga dalam pengembangan perangkat lunak, khususnya pengembangan game.

Selama proses pengembangan, kami berhasil mencapai beberapa hal penting, diantaranya:

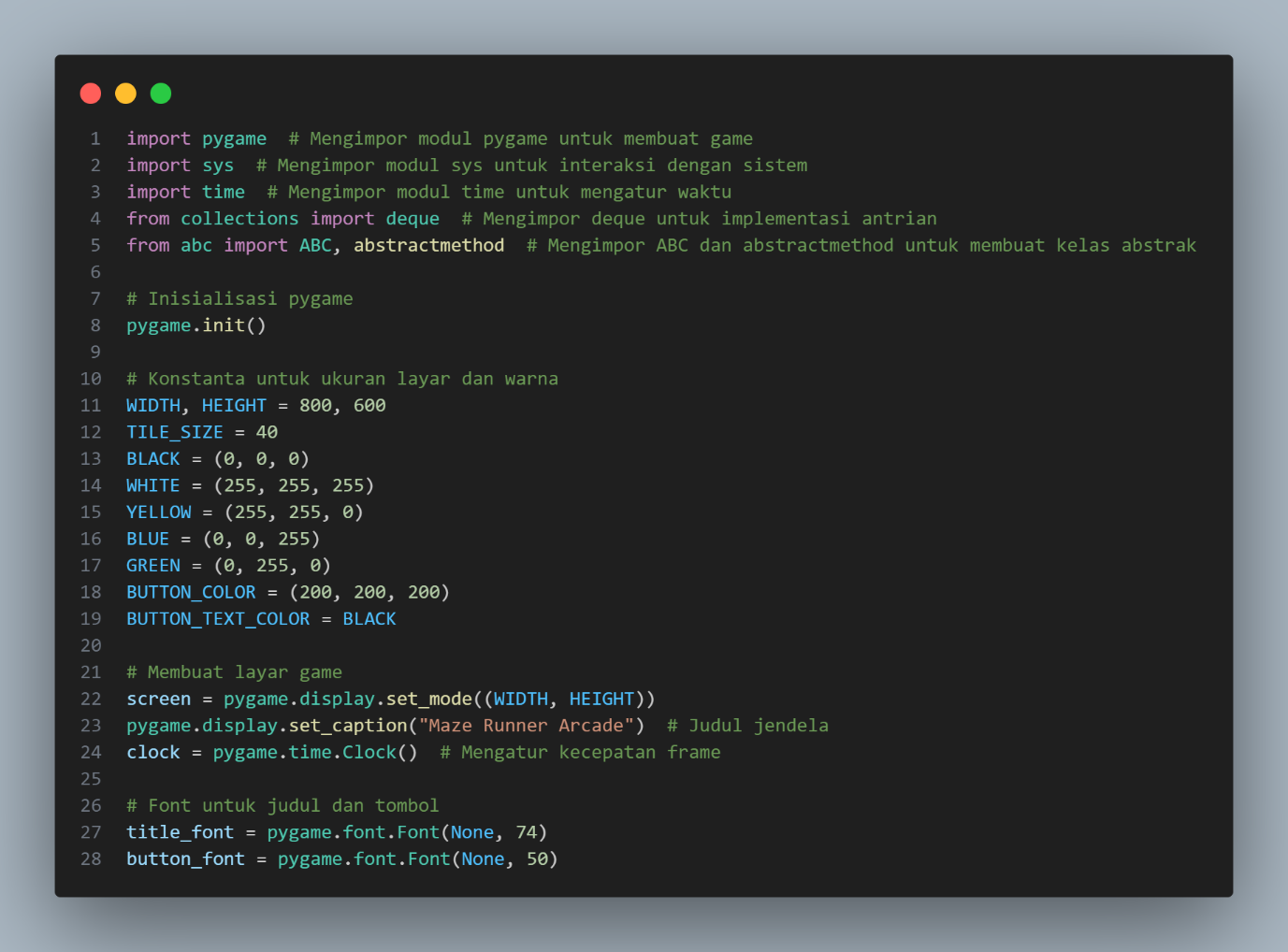
1. Mengimplementasikan konsep OOP (abstraksi, enkapsulasi, pewarisan, polimorfisme, dan komposisi) dengan baik dalam struktur kelas dan interaksi antar objek.
2. Membangun *game* 2D yang interaktif dan fungsional, yang melibatkan pergerakan pemain, *non-player character*, pengumpulan objek, dan penentuan kondisi menang/kalah.
3. Memanfaatkan pustaka Pygame dengan baik untuk mengimplementasikan elemen grafis, *input*, dan *game loop*.
4. Menerapkan algoritma BFS untuk pergerakan *guard* sehingga memberikan tantangan pada pemain.
5. Membuat *menu* utama, *menu* kesulitan level, dan *menu* akhir sebagai navigasi dan *state* pada *game*.

Melalui proyek ini, kami belajar banyak hal, baik dari aspek teknis maupun non-teknis. Kami mendapatkan pemahaman yang lebih mendalam mengenai konsep OOP dan implementasinya dalam *code*. Kami juga mendapatkan pengalaman dalam penggunaan pustaka Pygame dan bagaimana merancang *game loop*. Kami juga belajar tentang pentingnya perencanaan yang baik, manajemen waktu yang efektif, serta komunikasi dan koordinasi yang kuat dalam kerja tim. Kami belajar menghadapi dan mengatasi tantangan yang muncul selama proses pengembangan, serta belajar pentingnya *debugging* dan pengujian sistem.

Secara keseluruhan, proyek 'Maze Runner Arcade' telah berhasil memenuhi tujuan yang telah ditetapkan dan memberikan kami pembelajaran yang sangat berharga. Kami telah berhasil mengaplikasikan konsep OOP, membangun *game* yang fungsional, dan meningkatkan keterampilan kami dalam pengembangan perangkat lunak, khususnya pengembangan *game* 2D. Kami menyadari bahwa proyek ini masih memiliki keterbatasan dan masih ada area yang dapat ditingkatkan di masa mendatang. Namun, kami bangga dengan pencapaian yang telah kami raih dan yakin bahwa proyek ini memberikan landasan yang kuat bagi kami untuk mengembangkan aplikasi yang lebih kompleks dan inovatif di masa depan. Dengan demikian, proyek ini telah berhasil diselesaikan dan memberikan manfaat yang signifikan dalam meningkatkan pemahaman kami mengenai OOP dan pengembangan *game*.

# **Lampiran**

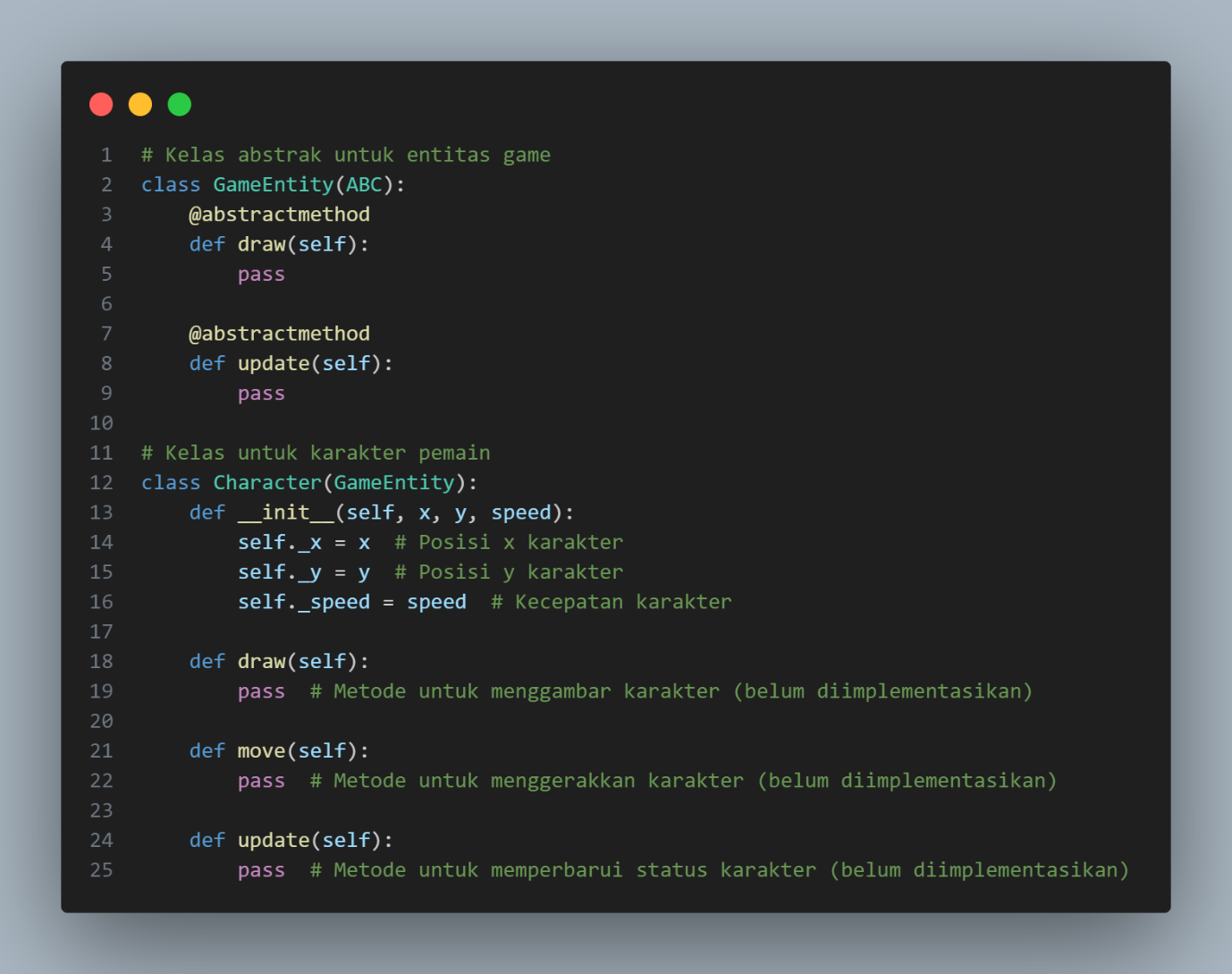
## **Lampiran 1 : Source Code Proyek**



**Gambar 7.2** Inisialisasi

**Deskripsi**:

Bagian ini berisi import modul, inisialisasi Pygame, deklarasi konstanta (ukuran layar, warna, ukuran tile), pengaturan layar game, caption, dan clock , serta pengaturan font.



**Gambar 7.3** Kelas abstrak Game Entity, dan karakter

**Deskripsi:**

Bagian ini berisi deklarasi kelas abstrak GameEntity dan Character, yang menjadi dasar bagi semua entitas game dan karakter dalam *game*.



**Gambar 7.4** Kelas Maze

**Deskripsi**: Bagian ini berisi definisi kelas *`Maze`*, termasuk atribut layout labirin, posisi bintang, dan metode untuk menggambar labirin dan mengumpulkan bintang.

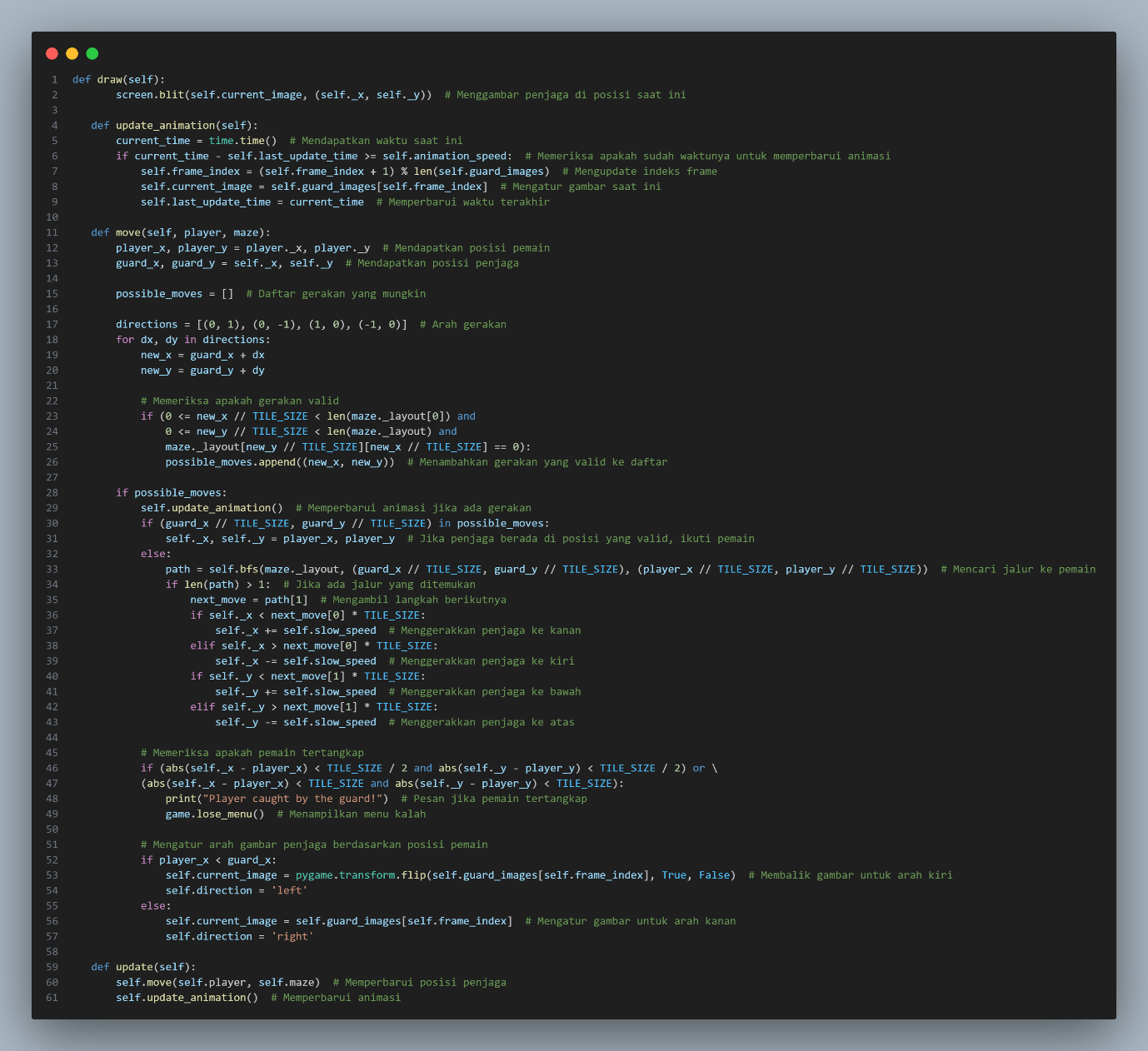


**Gambar 7.5** Kelas Player

**Deskripsi**: Bagian ini berisi kelas *`Player`* yang mengelola pergerakan pemain, animasi, interaksi dengan labirin dan bintang, serta kondisi menang.

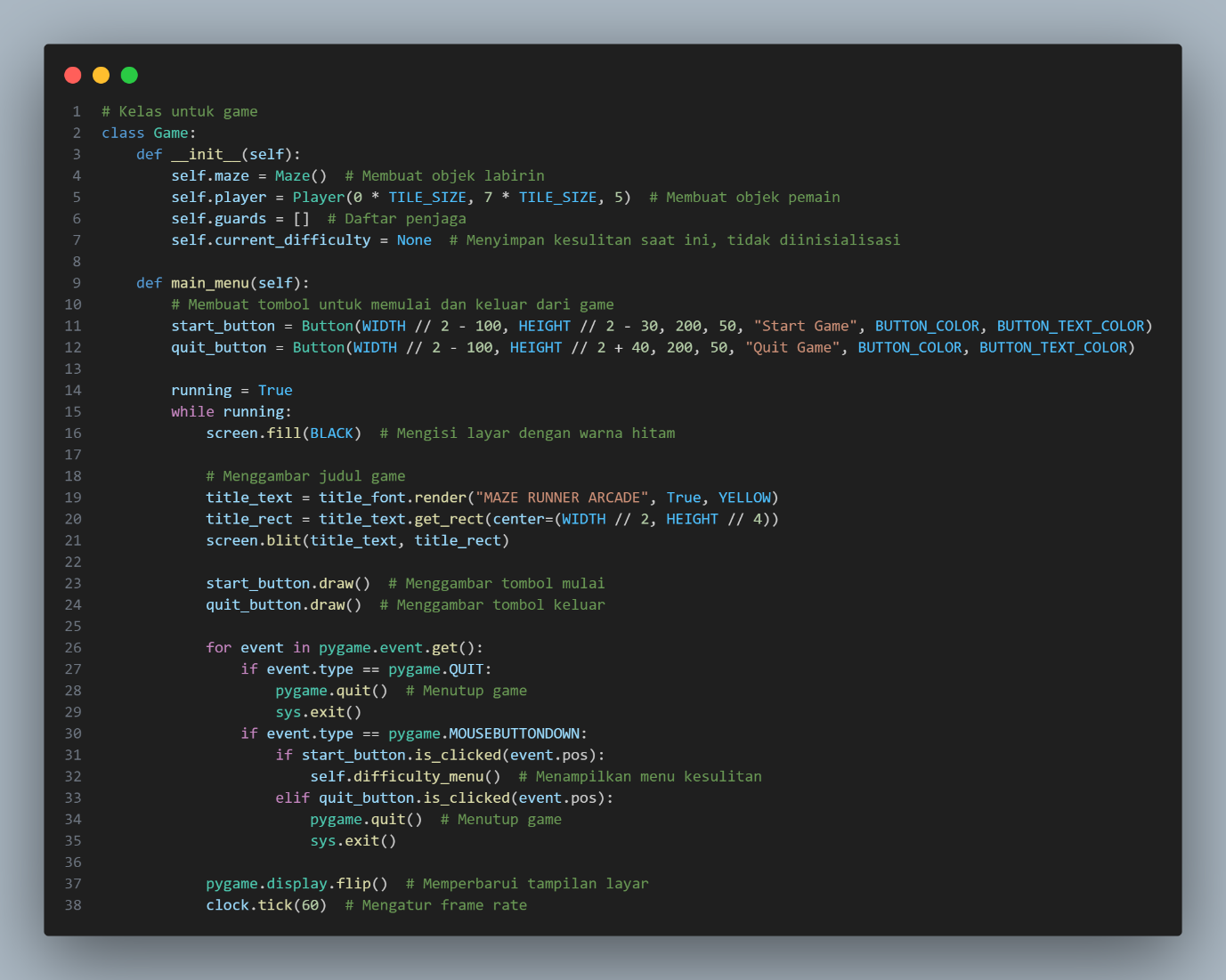


**Gambar 7.6** Kelas Button dan Guard (bagian 1)



**Gambar 7.7** Kelas Button dan Guard (bagian 2)

**Deskripsi**: Bagian ini berisi kelas *`Button`,* dan kelas *`Guard`* yang mengelola pergerakan dan animasi karakter *guard* termasuk algoritma BFS untuk mengejar pemain.



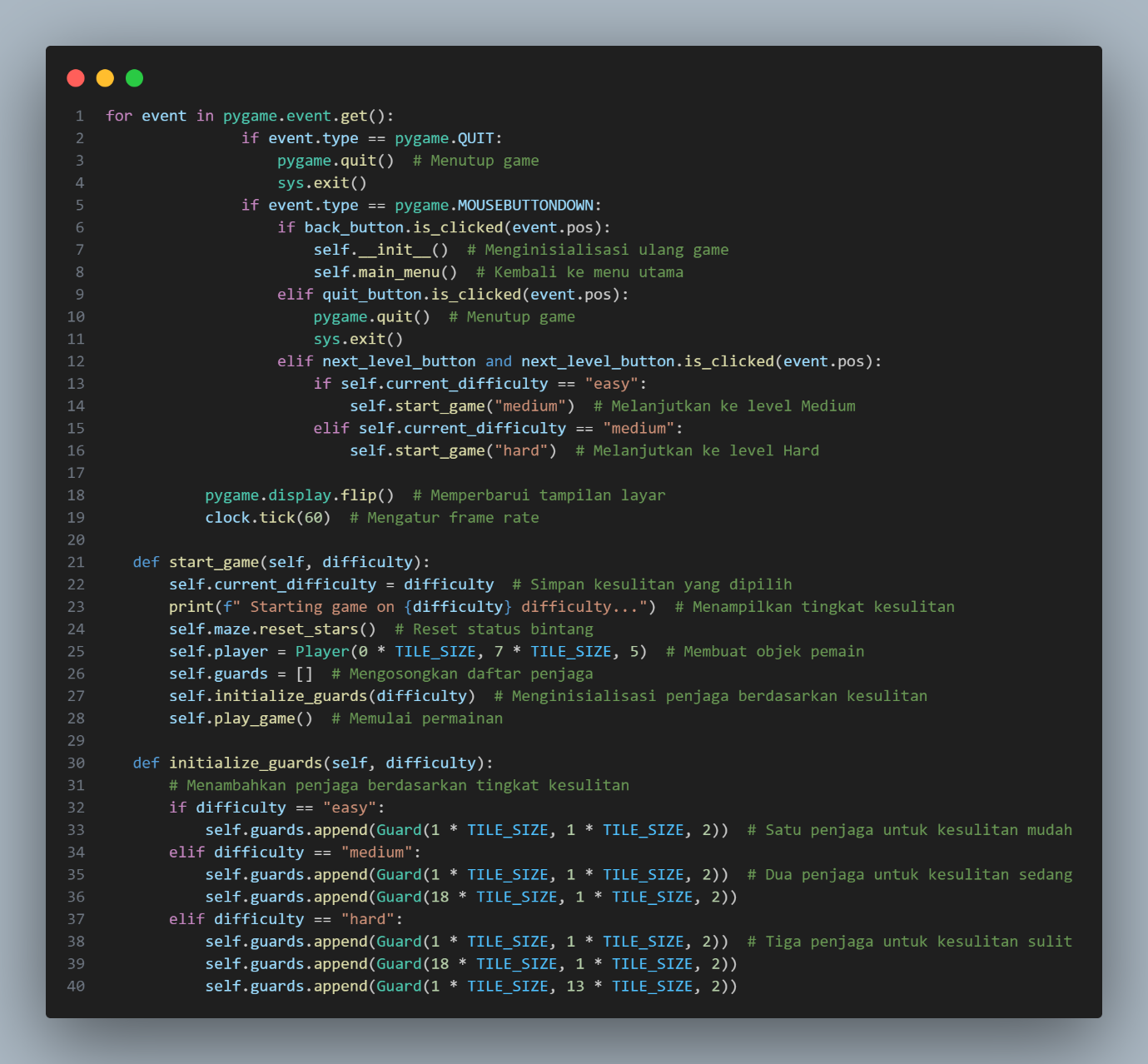
**Gambar 7.8** Kelas Game (Bagian 1)

**Deskripsi**: Bagian ini berisi kelas *`Game`*, khususnya bagian *`\_\_init\_\_`* dan *`main\_menu()`.* Kodingan ini mengelola inisialisasi *game*, menampilkan *menu* utama dan mengatur *event* yang terjadi pada *menu* tersebut.



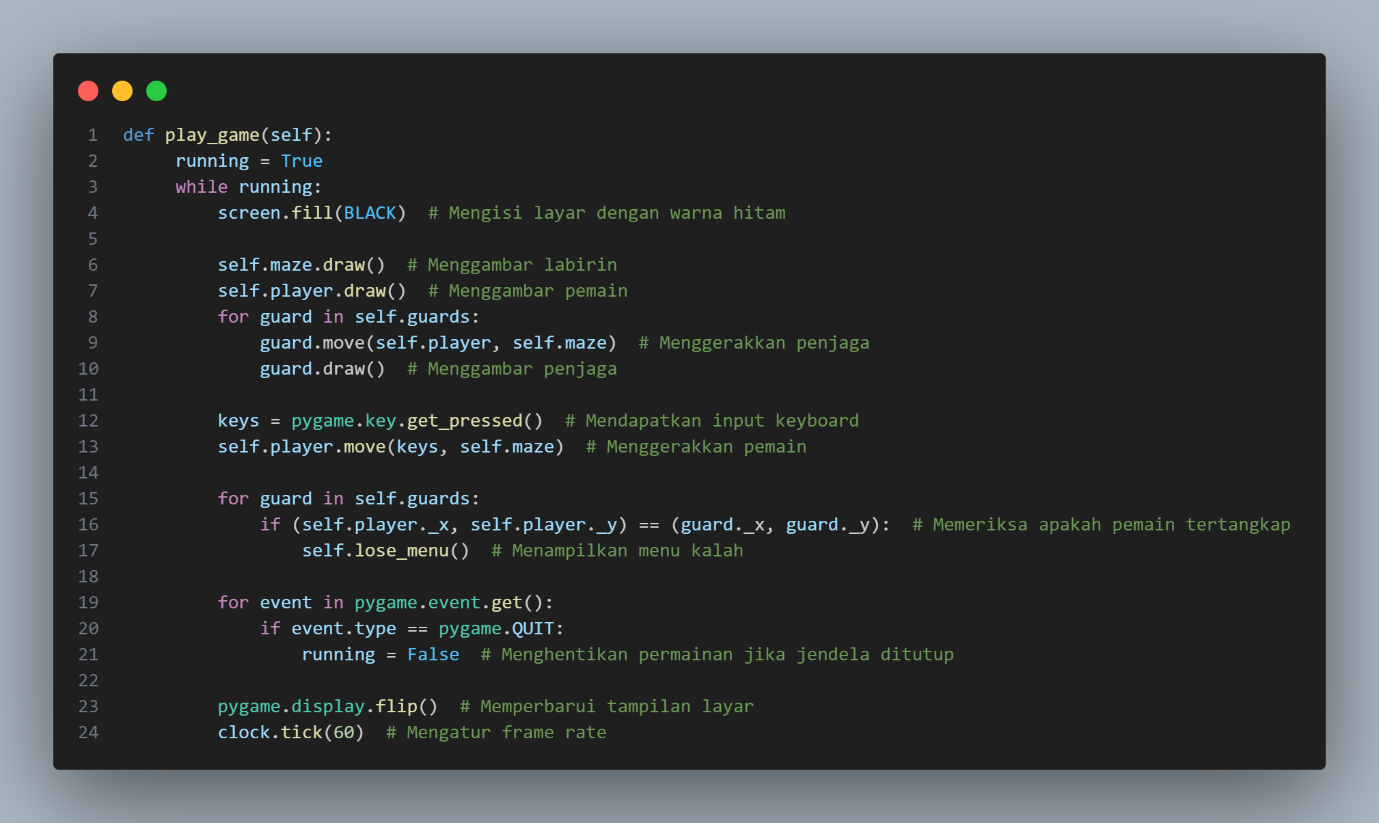
**Gambar 7.9** Kelas Game (Bagian 2)

**Deskripsi**: Bagian ini berisi kelas *`Game`,* khususnya metode *`finish\_menu()`.* Metode ini mengatur tampilan dan interaksi pada menu akhir setelah pemain memenangkan game, termasuk teks *"Congratulations!"*, tombol "*Next Level"* , *"Back to Main Menu"* dan *"Quit Game".*



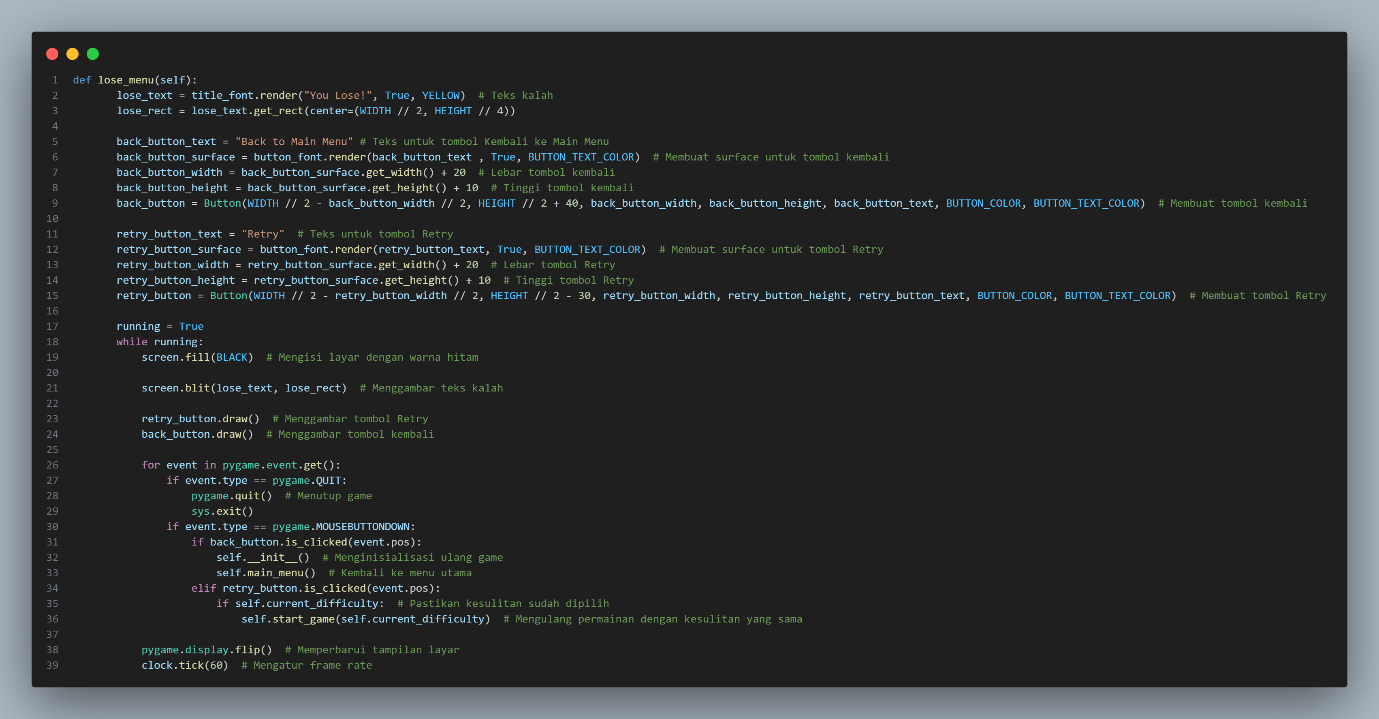
**Gambar 7.10** Kelas Game (Bagian 3)

**Deskripsi:** Bagian ini berisi kelas *Game*, khususnya metode *finish\_menu(), start\_game()* dan*initialize\_guards(),* yang bertanggung jawab untuk mengatur *menu* akhir (setelah menang), memulai *game* dengan tingkat kesulitan tertentu, dan menginisialisasi *guard*.



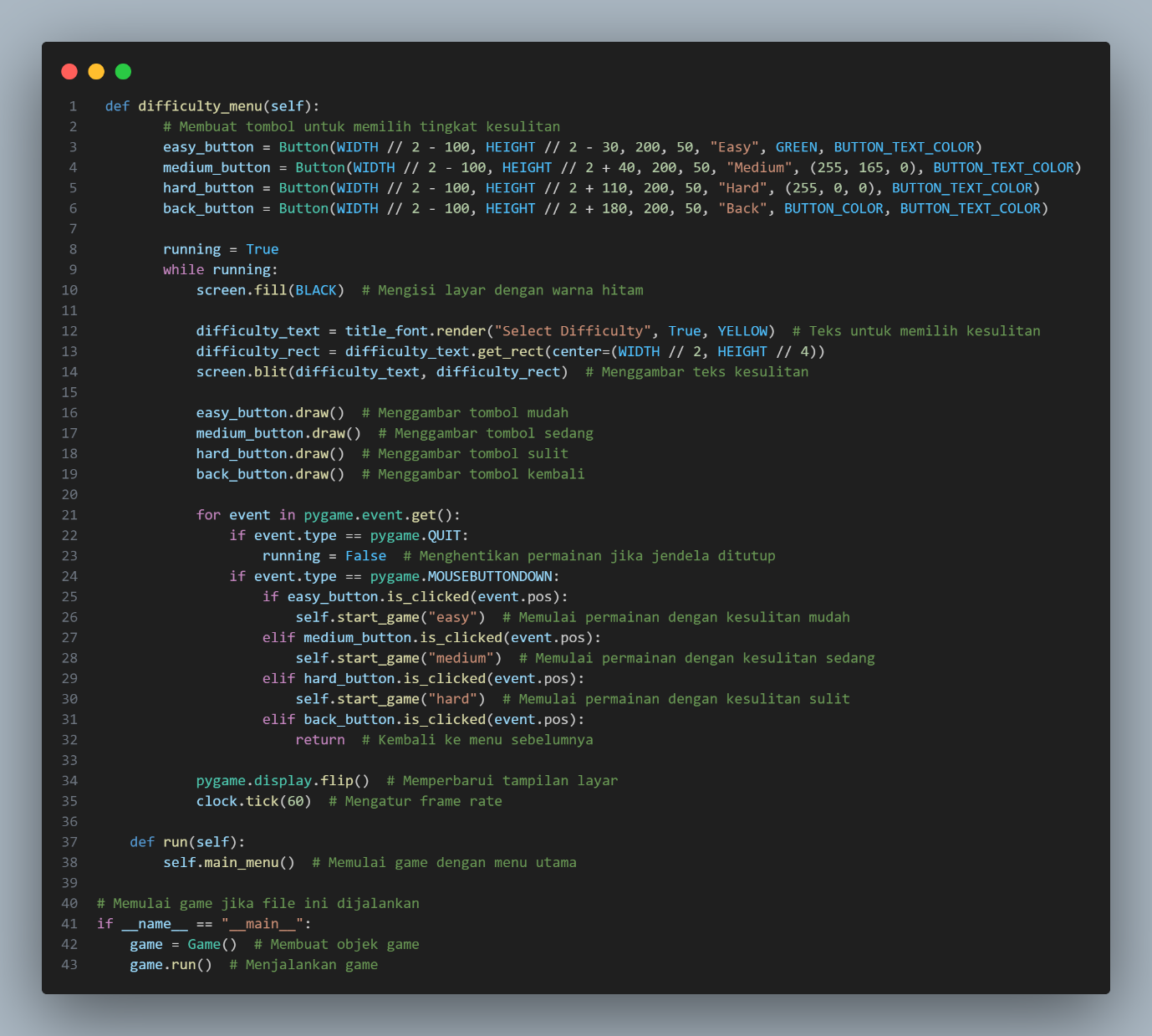
**Gambar 7.11** Kelas Game (Bagian 4)

**Deskripsi**: Bagian ini berisi kelas *Game*, khususnya metode *play\_game(),* yang merupakan inti dari *game loop*. Di dalam metode ini, labirin, pemain, dan guard akan digambar dan di-*update*.



**Gambar 7.12** Kelas Game (Bagian 5)

**Deskripsi**: Bagian ini berisi kelas *`Game`,* khususnya metode *`lose\_menu()`* yang mengatur tampilan ketika pemain kalah.



**Gambar 7.13** Kelas Game (Bagian 6) dan Run

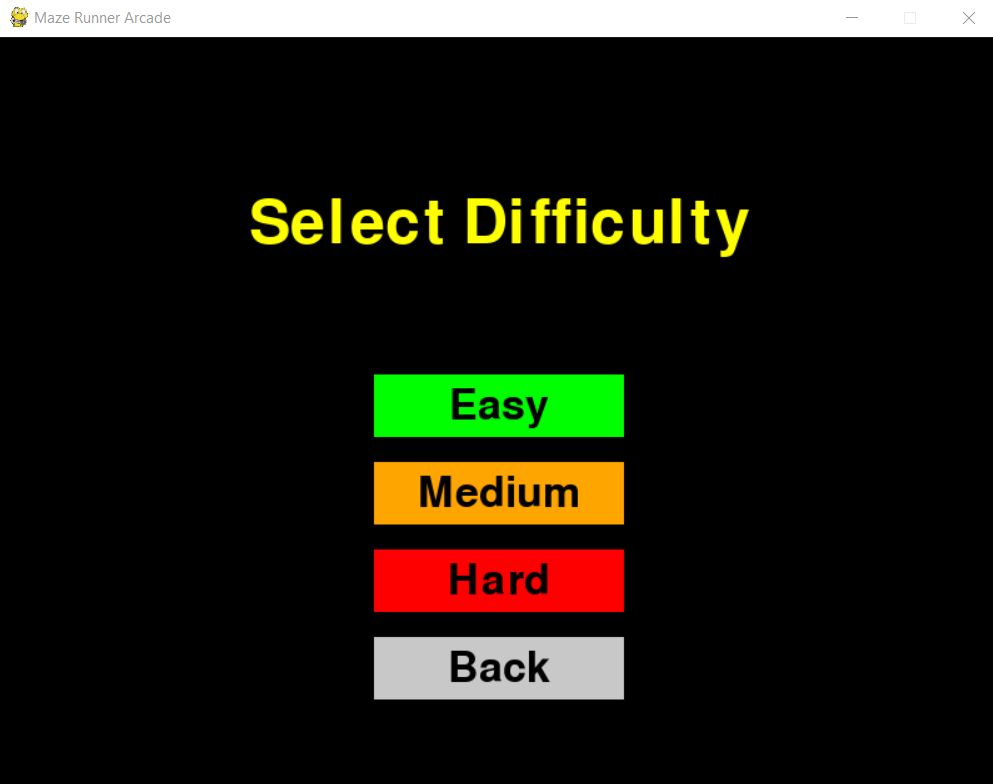
Deskripsi: Bagian ini berisi kelas *`Game`*, khususnya metode *`difficulty\_menu()`* yang mengatur menu pemilihan tingkat kesulitan dan *`run()`* untuk menjalankan *game*, dan main program yang membuat objek *game* dan menjalankan *game.*

## **Lampiran 2 : Tampilan Game *Maze Runner Arcade***



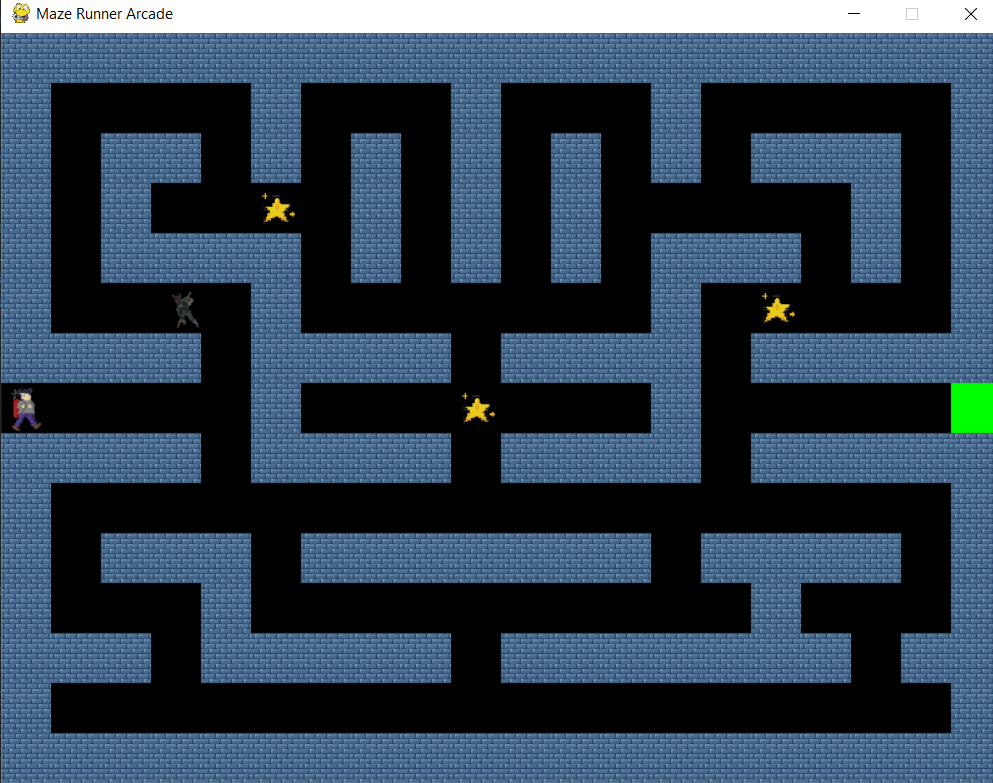
**Gambar 7.14** Home Screen

**Deskripsi** : Pada bagian home screen atau tampilan pada game ini, pemain akan disuguhkan dengan dua pilihan yaitu *Start Game* dan *Quit Game*. Jika, pemain memilih *Start Game*, maka, selanjutnya pemain akan dihadapkan dengan pilihan level yang mana seperti pada gambar setelah ini, dan jika pemain memilih *Quit Game*, maka game akan tertutup atau keluar.



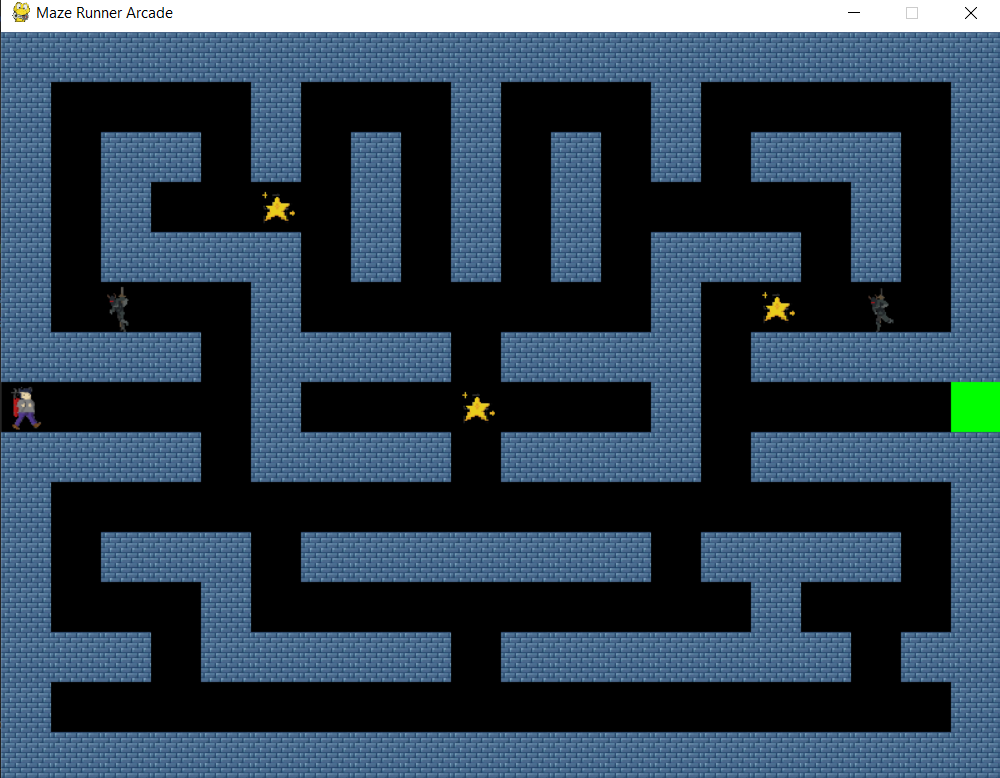
**Gambar 7.15** Menu Memilih Level

**Deskripsi** : Pada bagian menu memilih level ini para pemain akan diberikan 4 pilihan yaitu diantaranya pilihal level *easy*, *medium*, dan *hard*, selain itu juga terdapat tombol *back* jika pemain ingin ke menu awal.



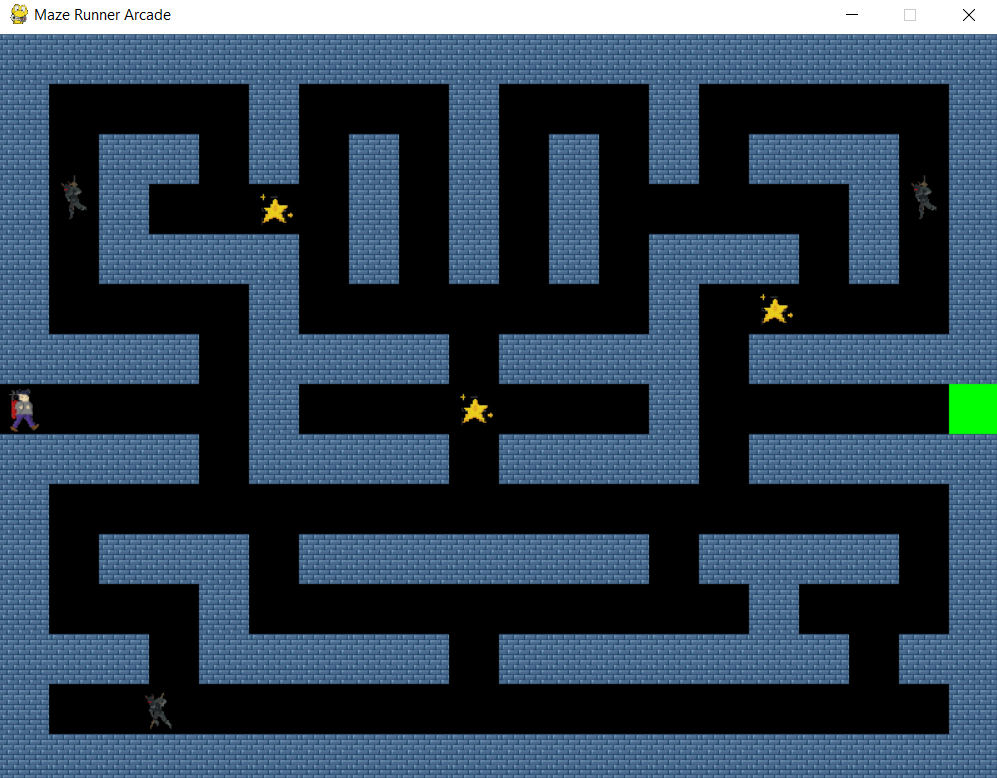
**Gambar 7.16** Tampilan Game di Level Easy

**Deskripsi** : Pada level game easy kurang lebih tampilannya seperti di atas, yaitu hanya terdapat satu penjaga di dalamnya.



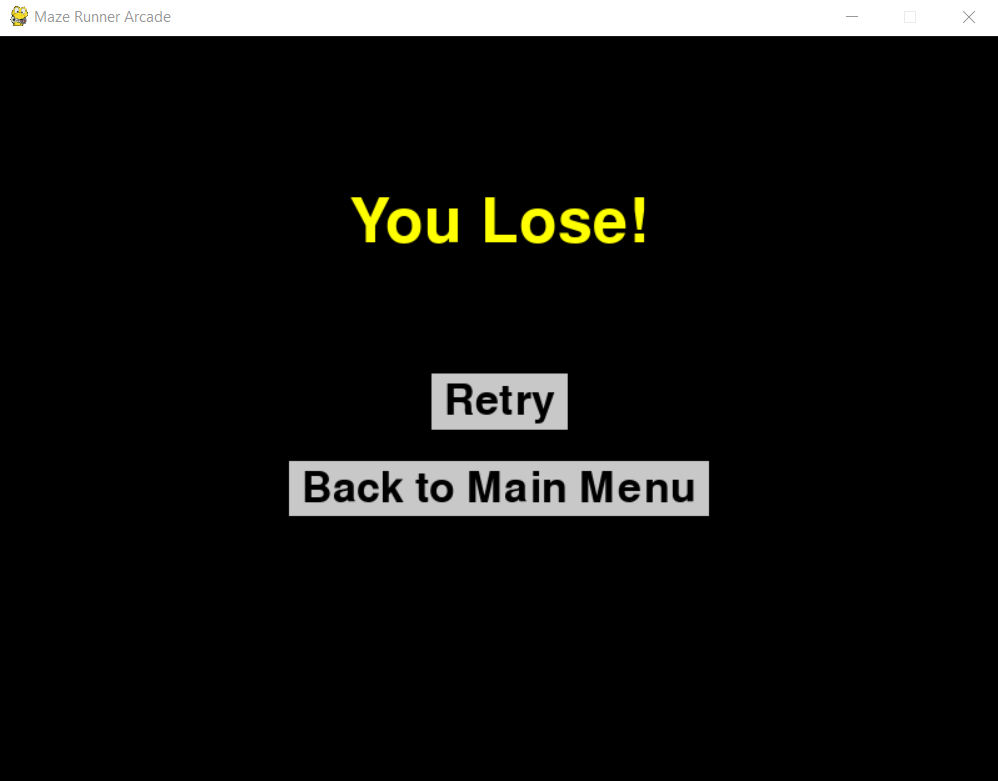
**Gambar 7.17** Tampilan Game di Level Medium

**Deskripsi** : Pada level medium sebenarnya tidak ada beda nya dengan level easy, yang membedakan hanya di jumlah penjaga nya yaitu ada dua.



**Gambar 7.18** Tampilan Game di Level Hard

**Deskripsi** : Pada level game Hard tampilan dan game play nya sama seperti kedua level sebelumnya, yang membadakan hanyalah di jumlah penjaga dan kecepatan penjaga yang mengejarnya.



**Gambar 7.19** Tampilan Ketika Kalah

**Deskripsi** : Ketika pemain tidak berhasil menyelesaikan level atau kalah, pemain akan dihadapkan oleh dua pilihan yaitu *retry* untuk mengulang game tersebut atau *back to main menu* yang nantinya akan Kembali ke tampilan awal.



**Gambar 7.20** Tampilan Game Ketika Menang

**Deskripsi** : Bagian yang terakhir adalah bagian ketika player atau pemain memenangkan game, nantinya pemain akan disuguhkan dengan tiga menu yaitu *next level* untuk lanjut ke level selanjutnya, *back to main menu* untuk kembali ke menu awal, dan *quit game* untuk keluar dari game.