

Tugas Besar

Merupakan sebuah game yang berjenis narrative-text-based-rpg. Berlatar tempatkan di sebuah kampus. Spesifiknya bertempat di ruang prodi elektro. Memiliki beberapa kasus event di dalam game tersebut, Player harus mencapai objective yang ada dan jangan sampai kewarasan player habis. Sistem permainan berfokus pada status "Kewarasan" (Sanity). Setiap keputusan yang salah atau kejadian random (RNG) dapat mengurangi poin kewarasan. Jika poin mencapai nol, permainan berakhir (Game Over).

Program ini mengaplikasikan pendekatan OOP(*Object-Oriented Programming*). Memiliki berbagai class serta ada salah satu class yang meng-*inheritance*. Ada juga beberapa fungsi atau prosedur yang telah dibuat untuk mempermudah membaca *code*.

Berikut ini adalah beberapa class yang terdapat di dalam program:

- **Class Player**

Mempresentasikan entitas pengguna atau pemain. Di dalamnya ada beberapa atribut dan method.

Atribut:

- **int kewarasan**
Menyimpan nilai kewarasan (HP) dari pemain yang dengan isi mulai dari 0 sampai 100.
- **bool adaKunci**
Sebuah flag inventory atau penanda di mana mengecek apakah user memiliki item kunci.
- **bool isMenyerah**
Status untuk keluar dari permainan secara manual yang dilakukan oleh player.

Method:

- **Constructor (Player())**
Mengisi kewarasan menjadi 100, adaKunci menjadi false, dan isMenyerah menjadi false. Ini adalah constructor untuk mengisi semua atribut dengan default value ketika object dibuat.
- **bool cekWaras()**
Sebuah fungsi yang mengembalikan nilai boolean untuk mengecek apakah player masih waras.

- `kewarasanDamage(int amount)`
Prosedur yang melakukan pengurangan sesuai dengan masukan dari argumen yang diberikan terhadap atribut kewarasan dengan diberikan visual text merah.
- `healKewarasan(int amount)`
Prosedur yang melakukan penambahan sesuai dengan masukan dari argumen yang diberikan terhadap atribut kewarasan dengan diberikan visual text hijau. Kewarasan telah diatur agar tidak bisa melebihi dari 100%.

- **Class Cerita**

Merupakan *Parent Class* yang menjadi kerangka dasar bagi setiap lokasi atau *scene* dalam game. Class inilah yang akan meng-*inheritance* tiap-tiap lokasi kedepannya. Di sinilah konsep Polymorphism diterapkan, bisa terlihat dari pemakaian *virtual function*.

Method:

- `virtual Cerita* mainkan(Player* p)`
Fungsi virtual ini dirancang untuk di-override oleh class turunannya. Fungsi ini mengembalikan pointer ke objek Cerita selanjutnya, memungkinkan perpindahan lokasi secara dinamis.

- **Class RuangProdi**

Merupakan *child class* yang menerapkan konsep pewarisan dari Class Cerita.

Atribut:

- `bool cekStatusAlur`
Status yang akan diberikan kepada RuangProdi, dimana ini akan membantu memberikan narasi intro di RuangProdi. Di mana ketika ini bernilai true, maka teks narasi yang muncul adalah narasi intro, dan ketika bernilai false akan diberikan teks narasi yang berbeda.

Method:

- Constructor (`RuangProdi(bool statusPertama)`)
Mengambil argumen yang diberikan untuk dimasukkan ke dalam atribut `cekStatusAlur`.
- `Cerita* mainkan(Player* p)` override
Fungsi dari *parent class* yang ditimpa atau *override*. Di sini RuangProdi sebagai lokasi awal, yang memiliki logika percabangan arah untuk lokasi selanjutnya.

- **Class LorongKiri**

Merupakan *child class* yang menerapkan konsep pewarisan dari Class Cerita.

Method:

- Cerita* mainkan(Player* p) override
Fungsi dari *parent class* yang ditimpa atau *override*. Di sini LorongKiri sebagai lokasi memakai konsep *RNG* untuk mendapatkan *Random Event* yang memicu elemen horror.

- **Class LorongKanan**

Merupakan *child class* yang menerapkan konsep pewarisan dari Class Cerita.

Method:

- Cerita* mainkan(Player* p) override
Fungsi dari *parent class* yang ditimpa atau *override*. Di sini LorongKanan sebagai lokasi teka-teki simple, yang dimana player bisa saja mencari kunci atau memaksa lanjut. Setelah player berhasil melewati lokasi ini, player akan langsung mencapai ending cerita.

- **Class LorongPanjang**

Merupakan *child class* yang menerapkan konsep pewarisan dari Class Cerita.

Method:

- Cerita* mainkan(Player* p) override
Fungsi dari *parent class* yang ditimpa atau *override*. Di sini LorongPanjang sebagai lokasi pencarian barang, lebih tepatnya menemukan item kunci yang dibutuhkan pada LorongKanan.

- **Class Parkiran**

Merupakan *child class* yang menerapkan konsep pewarisan dari Class Cerita.

Method:

- Cerita* mainkan(Player* p) override
Fungsi dari *parent class* yang ditimpa atau *override*. Class ini hanya sekedar dideklarasikan, tapi isinya masih kosong. Class Parkiran di desain atau *prototype* ketika user mencapai ending dan akan ada beberapa fitur yang bisa ditambahkan.

- **Class Game**

Class ini bertindak sebagai *Game Engine* sederhana atau pengontrol utama alur permainan. Class ini membungkus logika looping permainan, manajemen memori (pembuatan dan penghapusan objek), serta inisialisasi awal permainan.

Atribut:

- **Player* player**
Pointer yang menunjuk ke objek Player. Digunakan pointer agar objek bisa dibuat ulang (re-instantiated) saat pemain memilih untuk retry.
- **bool isRunning**
Status loop utama permainan. Jika bernilai false, maka loop permainan akan berhenti.
- **Cerita* lokasiAktif**
Pointer yang menyimpan lokasi (objek turunan Cerita) di mana player sedang berada saat ini.

Method

- **Constructor (Game())**
Menginisialisasi isRunning menjadi true dan pointer player menjadi null.
- **Destructor (~Game())**
Bertugas melakukan pembersihan memori terakhir saat aplikasi ditutup, seperti menghapus objek player dan mematikan semua suara.
- **start()**
Prosedur utama yang berisi *Game Loop*. Di sinilah logika *retry* (mengulang permainan), perpindahan lokasi (polymorphism), pengecekan kondisi Game Over, dan pemanggilan fungsi visual (intro, outro) dijalankan secara prosedural.

Berikut ini adalah beberapa fungsi/prosedur dari masing-masing yang terdapat di dalam program:

- **delay(int ms)**
Prosedur yang berisikan fungsi sleep(), yang dimana memiliki parameter ms. Parameter ms dimasukkan ke dalam fungsi sleep(). Prosedur ini ditugaskan untuk mengatur delay saat program dijalankan
- **aturWarna(int colorCode)**
Prosedur yang berisikan fungsi SetConsoleTextAttribute(), menerima parameter sebagai colorCode. Parameter colorCode akan dimasukkan ke dalam fungsi

`SetConsoleTextAttribute()`. Prosedur ini ditugaskan untuk mengatur warna dari teks di dalam cmd.

- `int rollAngka()`

Fungsi yang mengembalikan nilai dari fungsi `rand()`. Yang di mana fungsi `rand()` sudah diatur agar hanya bisa mengeluarkan akhiran dari angka 1-100. Setelah mendapatkan hasil dari fungsi `rand()`, maka akan di return dengan tipe data `int`.

- `salahOpsi()`

Prosedur yang bertugas untuk mencetak sebuah text setiap kali dipanggil. Di dalamnya memanggil `aturWarna()` untuk memberikan warna text. Lalu text ini untuk memberikan sebuah text himbauan ketika user memberikan pilihan yang tidak sesuai.

- `slowPrint(string text, int speed = 40)`

Memiliki dua parameter, `text` dan `speed`. Dimana parameter `text` dengan tipe data `string` yang wajib diisi, dan parameter `speed` dengan tipe data `int` yang opsional karena memiliki default value 40. `slowPrint()` bertujuan agar setiap dipanggil, `text` akan keluar sesuai dengan kecepatan yang diberikan. Berbeda dengan `cout` yang dimana langsung di print secara instan.

- `bukaFoto(string namaFile)`

Prosedur ini bertujuan untuk membuka sebuah file, lebih tepatnya gambar. Menggunakan fungsi `system()` untuk membuka filenya dan mengambil variabel `namaFile` sebagai tujuan filenya.

- `textHantu()`

Prosedur ini bertujuan untuk mencetak sebuah file `.txt` dan membuka sebuah gambar yang sudah ada. Di sini library `<fstream>` digunakan. Prosedur ini terpanggil ketika player mengalami game over.

- `textEnding()`

Prosedur ini bertujuan untuk mencetak sebuah file `.txt` dan membuka sebuah gambar yang sudah ada. Di sini library `<fstream>` digunakan. Prosedur ini terpanggil ketika player telah mencapai akhir cerita.

- `playBGM(string namaFile)`

Prosedur ini memiliki parameter `namaFile` dengan tipe data `string`. Disini library `<mmsystem.h>` digunakan. Prosedur ini bertujuan untuk memulai sebuah lagu dibelakang

layar selama program dijalankan. Parameter namaFile dipakai untuk dimasukkan ke dalam fungsi library <mmsystem.h>

- **pauseBGM()**
Prosedur ini memanggil fungsi yang ada di library <mmsystem.h>. Bertujuan untuk mem-*pause* lagu yang sudah di-*play* sebelumnya.
- **resumeBGM()**
Prosedur ini memanggil fungsi yang ada di library <mmsystem.h>. Bertujuan untuk memulai kembali lagu yang di-*pause* sebelumnya, lanjut dari durasi sebelum di-*pause*.
- **stopBGM()**
Prosedur ini memanggil fungsi yang ada di library <mmsystem.h>. Bertujuan untuk menghentikan lagu yang telah di setel.
- **playSFX(string namaFile)**
Prosedur ini memiliki parameter namaFile dengan tipe data string. Prosedur ini memanggil fungsi playSoundA dari library <windows.h>. Bertujuan untuk memutar *sound effect* dengan membuka file yang mengarah ke namaFile.
- **tampilMainMenu()**
Prosedur ini menyetak text yang akan ditampilkan pertama kali setelah program dijalankan. Bertujuan untuk memberikan menu kepada user ketika program dibuka.
- **tampilCredits()**
Prosedur ini menyetak text yang akan ditampilkan setelah permainan berakhir dan sebelum program di hentikan. Bertujuan untuk menampilkan informasi dan memberikan apresiasi.
- **tampilGameOver()**
Prosedur ini menyetak text visual dan narasi horor ketika user kalah (kewarasan habis). Di dalamnya juga memanggil prosedur textHantu() untuk memicu efek jumpscare
- **tampilEnding()**
Prosedur ini menyetak text yang berisikan cerita ketika user sudah berada di penghujung cerita. Di dalam prosedur inilah tampilCredits() dipanggil.
- **int tampilMenuRetry()**

Fungsi ini berfungsi untuk mencetak sebuah menu, di mana menu ini memberikan informasi kepada player apakah mau mengulang program menjadi semula atau keluar dari program. Opsi yang dipilih player akan menjadi nilai return.