BAB IV

HASIL DAN IMPLEMENTASI

IV.1. Hasil

Adapun hasil Perancangan Penerapan Algoritma Greedy Untuk Pergerakan *Ghost* Pada Permainan Pac-Man. yang sudah dibuat, dapat dilihat di bawah ini pada bab ini.

IV.1.1. Kebutuhan Perangkat Keras *(Hardware)* dan Perangkat Lunak *(Software)*

Dalam penerapan sistem yang dibuat tidak terlepas dari perangkat keras dan perangkat lunak. Untuk menguji program atau sistem informasi, digunakan komputer dengan spesifikasi sebagai berikut:

A. Perangkat Keras (*Hardware*)

1. Laptop
2. Minimal Intel (R) Core 2
3. RAM 1 GB
4. Hardisk Minimal 500 GB
5. *Keyboard*

B. Perangkat Lunak (*Software*)

1. Sistem Operasi Microsoft Windows (*Seven*)
2. Bahasa Pemrograma Java
3. Editor Netbeans

IV.2. Implementasi Sistem

Implementasi sistem adalah prosedur yang dilakukan untuk menyelesaikan desain yang ada dalam dokumen desain sistem yang disetujui dan menguji, menginstal, memulai, serta menggunakan sistem yang baru atau sistem yang diperbaiki.

Penggunaan suatu komputer untuk pemecahan masalah membutuhkan suatu sistem yang baik, sehingga memungkinkan berhasilnya komputer dalam melaksanakan tugasnya, yaitu mengolah data menjadi informasi. Langkah implementasi yang dilakukan dalam Penerapan Algoritma Greedy Untuk Pergerakan *Ghost* Pada Permainan Pac-Man, Menyediakan perangkat keras *(Hardware)* dan perangkat lunak *(Software)*. Dalam tahap ini disediakan perangkat keras.

Perangkat lunak yang dibutuhkan adalah Sistem Operasi XP dan bahasa pemograman yang digunakan untuk menulis program ke dalam komputer. Menguji sistem menjelaskan mengenai hasil pengujian sistem yang dilakukan pada Penerapan Algoritma *Greedy* Untuk Pergerakan *Ghost* Pada Permainan Pac-Man.

Metode pengujian sistem yang digunakan adalah *black-box testing*. *Black-box testing* adalah metode pengujian yang dimana penilaian terhadap sebuah aplikasi bukan terletak pada spesikasi logika/fungsi aplikasi tersebut, tapi masukan (*input)* dan keluaran (*output)*. Dengan berbagai masukan (*input)* yang diberikan akan dievaluasi apakah suatu sistem/aplikasi dapat memberikan

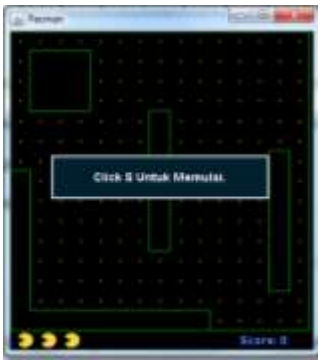
keluaran (*output*) yang sesuai dengan harapan penguji. Pengujian sistem dilakukan dengan cara sebagai berikut:

1. Hasil pengujian sistem disajikan dalam bentuk tabel.
2. Pengujian ditargetkan pada setiap proses yang dimiliki Penerapan Algoritma Greedy Untuk Pergerakan *Ghost* Pada Permainan Pac-Man

IV.3. Pembahasan Hasil

1. Form Utama

Sewaktu menjalankan program *form splash screen* akan muncul pertama sekali sebagai *loading* menuju *form* utama, pada *form* utama ini aplikasi *games* pacman belum bisa dijalanan langsung, *user* sebelumnya harus menekan tombol "s" sebagai dimulainya permainan *games* pacman, berikut tampilanya.



Gambar IV.1 *Form Splash Screen*

2. Form Interface

Form ini akan tampil sewaktu *user* sudah mengclick tombol "s" pada form sebelumnya sebagai tanda permainan *games* pacman sudah bisa dimulai, pada *form* ini terdapat 6 *ghost*, yang akan berjalan secara random yang sudah di setting sebelumnya di *sourcenya*, dan arah pacman terdiri dari 4 arah, yaitu arah keatas (tombol panah keyboard ke atas) , arah ke bawah (tombol panah *keyboard*

kebawah), arah kekanan (tombol panah keyboard ke kanan) dan arah ke kiri (tombol panah *keyboard* ke kiri), berikut :

Gambar IV.2 *Form Interface*

3. Uji Coba Program 1

1. *User* mengarahkan pacman untuk mencari nilai score 81:

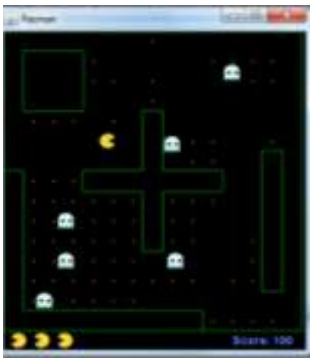
Pada ujicoba ini pacman sekarang berada ke arah atas, dimana pada uji coba ini, *score* yang sudah berhasil dikumpulkan *user* adalah 81, dimana score ini didapat setiap pacman memakan 1 titik maka *score* akan bernilai 1, tetapi jika pacman terkena oleh *ghost* maka permainan user akan kalah, oleh karena itu *user* harus menghindari *ghost* menabrak pacman.

Gambar IV.3. Score Games Pacman 1

4. Uji Coba Program 2

2. User mengarahkan pacman untuk mencari nilai score 100:

Pada ujicoba ini pacman sekarang berada ke arah atas, dimana pada uji coba ini, *score* yang sudah berhasil dikumpulkan *user* adalah 100, dimana score ini didapat setiap pacman memakan 1 titik maka *score* akan bernilai 1, tetapi jika pacman terkena oleh *ghost* maka permainan *user* akan kalah, oleh karena itu *user* harus menghindari *ghost* menabrak pacman.



Gambar IV.4. Score Games Pacman 1

5. Uji Coba Program 3

3. User mengarahkan pacman untuk mencari nilai score 118:

Pada ujicoba ini pacman sekarang berada ke arah atas, dimana pada uji coba ini, *score* yang sudah berhasil dikumpulkan *user* adalah 118, dimana *score* ini didapat setiap pacman memakan 1 titik maka *score* akan bernilai 1, tetapi jika pacman terkena oleh *ghost* maka permainan *user* akan kalah, oleh karena itu *user* harus menghindari *ghost* menabrak pacman.

Gambar IV.5. Score Games Pacman 1

6. Uji Coba Program 4

4. Pacman Tertabrak Ghost:

Pada ujicoba ini pacman tertabrak *ghost* sehingga *user* kalah dan *score* akan kembali berubah menjadi 0, dan pacman akan kembali ke tempat semula untuk memulai pergerakan baru lagi, pada permainan ini ghost akan bergerak secara random, jadi tidak dikendalikan oleh *user* tetapi sudah di *setting* oleh *souce code* java

Gambar IV.6. Pacman Tertabrak Ghost

IV.4. Pemeliharaan Sistem *(Maintenance)*

Pemeliharaan sistem informasi adalah suatu upaya untuk memperbaiki, menjaga, menanggulangi, mengembangkan sistem yang ada. Pemeliharaan ini di perlukan untuk meningkatkan efisiensi dan efektivitas kinerja sistem yang kita ada

agar dalam penggunaannya dapat optimal. Berikut ini beberapa pengertian lain tentang pemeliharaan sistem dari beberapa sumber :

- Merupakan siklus terakhir dari SDLC
- Pemeriksaan periodik, audit dan permintaan pengguna akan menjadi *source*

Untuk melakukan perawatan sistem diseluruh masa hidup sistem Pemeliharaan sistem merupakan cara terbaik untuk menjaga efiensi sistem yang sudah ada. Berikut merupakan beberapa alasan mengapa kita perlu memelihara sistem yang ada :

a. agar dapat meningkatkan kinerja sistem
b. Menyesuaikan dengan perkembangan, agar sistem yang ada tidak tertinggal

IV.5. Kelebihan Dan Kekurangan

1. Kelebihan

Strategi yang di gunakan algoritma *greedy* cukup ampuh untuk di terapkan pada persoalan-persoalan optimasi. Hasil solusi yang di dapatkan algoritma *greedy* cenderung mendekati atau menghampiri hasil solusi optimal, selain itu, yang menjadi kelebihan algoritma *greedy* sendiri adalah implementasinya yang sangat sederhana dan langsung.

2. Kekurangan

1. Algoritma *greedy* tidak beroperasi secara menyeluruh terhadap semua alternatif solusi yang ada (sebagaimana pada metode *exhaustive search*).
2. Pemilihan fungsi seleksi Mungkin saja terdapat beberapa fungsi seleksi yang berbeda, sehingga kita harus memilih fungsi yang tepat jika kita ingin algoritma bekerja

dengan benar dan menghasilkan solusi yang benar-benar optimum. Karena itu, pada sebagian masalah algoritma *Greedy* tidak selalu berhasil memberikan solusi yang benar-benar optimum.