ISSN 1410 - 9158

Majalah Ilmiah
Informatika, Komputer, dan Bisnis

FORMAT

Volume 8, Nomor 3, September 2007

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER AKAKOM
YOGYAKARTA

Majalah Ilmiah
Informatika, Komputer, dan Bisnis
FORMAT

PELINDUNG:
Ketua Yayasan Pendidikan Widya Bakti

KETUA UMUM:
Ketua STMIK AKAKOM Yogyakarta

KETUA DEWAN REDAKSI:
Bambang P. D. P., S.E., Akt., S.Kom., M.MSi.

ANGGOTA DEWAN REDAKSI:
Ir. F. Soesianto, B.Sc.E., Ph.D.
Prof. H. Adhi Susanto, M.Sc., Ph.D.
Drs. Tri Prabawa, M.Kom.
Ir. Surjono, M.Phil.
Ir. Sudarmanto, M.T.
Ir. M. Guntara, M.T.
Ir. Totok Suprawoto, M.M.
Budi Sugihardjo, S.E., M.M.
Heru Agus Triyanto, S.E., M.M.

REDAKTUR PELAKSANA:
Indra Yatini Buryadi, S.Kom., M.Kom.

SEKRETARIS:
Al. Agus Subagyo, S.E., M.Si.

LAYOUT dan PRODUKSI:
Dison Libardo, S.E., M.Kom.

SIRKULASI:
Totok Budioko, S.T.

DOKUMENTASI:
Dra. Torsinawati
Sukar

Majalah Ilmiah FORMAT diterbitkan empat bulan sekali oleh STMIK AKAKOM dengan ISSN 1410 – 9158

ALAMAT REDAKSI:
STMIK AKAKOM
Jl. Raya Janti, *Ring Road* Timur, Yogyakarta 55198
Telepon : 62-0274-486664
Faksimile : 62-0274-486438 E-mail : format@netexecutive.com

Dari Redaksi

Kami panjatkan puji dan syukur atas rahmat dan berkah dari Tuhan Yang Maha Esa hingga kami dapat menyelesaikan dan menerbitkan majalah Format pada nomor pertama, tahun pertama. Pada tahun yang pertama ini kami mencoba untuk memperluas cakupan materi dari berbagai hasil penelitian dan karya ilmiah, namun tetap sesuai dengan misinya.

Dalam edisi ini para pembaca akan melihat topik-topik mengenai *Penyelesaian Model Transportasi Dengan Algoritma Genetika, Sistem Komputerisasi Koreksi Dan Penilaian Uji Coba Ujian Nasional Menggunakan LJK Untuk Siswa SMA Dan MA, Asynchronus Javascript and XML (Ajax) : Membangun Web yang Responsif, Studi Kasus Pemilihan Mata Kuliah KRS Online, Algoritma Dan Implementasi Penyaringan Citra,, Datamining Dalam Kontrol Industri, Sistem Penjamakan Pada Komunikasi Serat Optik, Sistem Informasi Online Realtime, Agen : Teori Dan Implementasinya, Memanfaatkan Voice Modem untuk Mesin Penjawab Telepon*, yang sekira akan sangat menarik untuk diulas.

Harapan kami semoga apa yang kami suguhkan kali ini dapat membawa manfaat bagi peminat, dan menambah referensi pembaca pada bidang-bidang tertentu. Terima kasih diucapkan, atas saran dan masukan yang telah kami terima demi kemajuan majalah ilmiah ini. Saran, ide dan gagasan dari pembaca tetap kami tunggu untuk perbaikan pada penerbitan edisi mendatang di abad millenium ini.

Daftar Isi

AGEN: TEORI DAN IMPLEMENTASINYA

Oleh : Sari Iswanti

INTISARI

Istilah agen beberapa tahun terakhir ini banyak diperbincangkan. Bidang AI (*Artificial Intelligent*) merupakan salah satu bidang yang giat melakukan penelitian mengenai agen (paradigma agen). Agen memiliki pengertian segala sesuatu yang dapat dipandang menangkap/menanggapi lingkungannya melalui sensor dan bertindak terhadap lingkungannya melalui efektor. Dalam merancang sebuah agen perlu memperhatikan prinsip-prinsip perancangan agen sehingga agen memiliki unjuk kerja yang baik dan dapat menyelesaikan/mengerjakan pekerjaan dalam lingkungannya sesuai dengan tujuan yang diharapkan.

Tulisan ini membahas proses analisa dan perancangan sebuah agen dalam bentuk simulasi agen, yaitu simulasi kerja *vacuum cleaner*. Analisa dan perancangan agen dikerjakan dengan memperhatikan 4 hal yaitu *Percept, Action, Goal,* dan *Environment* (PAGE).

Agen yang berhasil dibuat dalam bentuk program agen ini masih banyak memiliki keterbatasan, salah satunya adalah adanya sampah yang tidak dapat terambil semua sehingga tujuan tidak dapat tercapai 100 %. Hal ini menyebabkan unjuk kerja dari *vacuum cleaner* tidak maksimal.

Kata kunci : agen, PAGE, simulasi, unjuk kerja

1. PENDAHULUAN

Istilah agen menjadi lebih terkenal seiring dengan perkembangan internet, meskipun sebenarnya pengertian agen tidak selalu terkait dengan internet. Paradigma agen beberapa dekade terakhir dipelajari/diteliti dalam bidang kecerdasan buatan (*Artificial Intelligent*) khususnya kecerdasan buatan yang terdistribusi (*Distributed Artificial Intelligent*).

Agen yang merupakan salah satu bidang aplikasi kecerdasan buatan, secara umum memiliki pengertian : segala sesuatu yang yang dapat dipandang menangkap/menanggapi lingkungannya melalui sensor dan bertindak terhadap lingkungannya

melalui efector [Norvig, 1995]. Sensor adalah bagian yang merangsang agen dan efektor adalah bagian yang digunakan oleh agen untuk melakukan tindakan. Jika dikaitkan manusia sebagai agen maka mata dan telinga dapat dianggap sebagai sensor dan efektornya antara lain dapat berupa tangan, kaki, lengan, dan mulut.

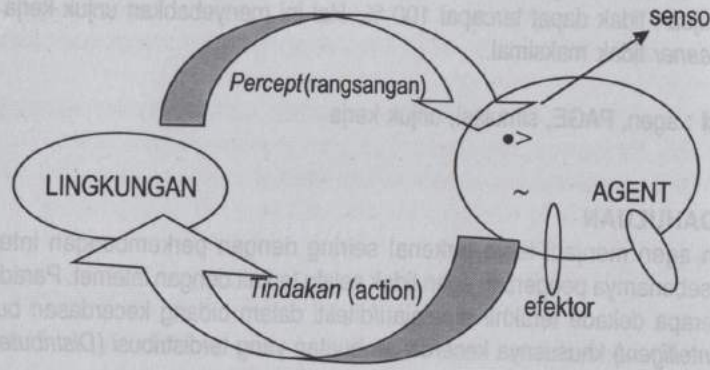
Dalam merancang sebuah agen perlu memperhatikan prinsip-prinsip merancang sebuah agen yang harus diikuti sehingga agen dapat menyelesaikan/ mengerjakan pekerjaan dalam lingkungannya dengan baik sesuai dengan tujuan yang diharapkan. Merancang sebuah agen diharapkan adalah agen yang rasional yaitu yang dapat melakukan tindakan dengan benar. Tindakan yang benar merupakan salah satu ukuran kesuksesan sebuah agen. Kriteria-kriteria untuk mengukur kesuksesan sebuah *agent* dituangkan dalam *performance measure* (ukuran performance).

Dalam tulisan ini akan dibahas proses analisa dan perancangan sebuah agen berupa *vacuum cleaner* dalam bentuk simulasi (*simulator vacuum cleaner*).

2. LANDASAN TEORI

2.1. Agen

Dari pengertian yang disampaikan di bagian terdahulu, secara umum agen dapat digambarkan seperti berikut :



Gambar 1. Interaksi agen dengan lingkungannya

Sebuah agen seharusnya adalah agen yang rasional yaitu agen yang dapat melakukan tindakan dengan benar. Tindakan yang benar merupakan tolok ukur kesuksesan sebuah agen. Kesuksesan sebuah agen dapat ditentukan berdasarkan kriteria tertentu dan kriteria tersebut berbeda-beda tergantung jenis agennya. *Performance Measure* (ukuran unjuk kerja) digunakan untuk menentukan kriteria-kriteria apa saja yang diharus dipenuhi untuk menyatakan bagaiman sebuah agen itu sukses.

Rasional pada agen tergantung pada 4 hal yaitu :

1. *Performance Measure* yang menyatakan derajat kesuksesan
2. Semua yang diterima oleh agen, dapat dilacak kembali (yang diterima oleh agen disimpan dalam serangkaian persepsi)
3. Apa saja yang diketahui agen tentang lingkungannya
4. Tindakan (*action*) apa saja yang akan dikerjakan oleh agen

Untuk merancang sebuah agen, hal yang sangat penting adalah merancang program agen (*agent program*) yaitu sebuah fungsi yang mengimplementasikan pemetaan (merubah) dari *percept* ke *action.* Diasumsikan bahwa program ini akan dapat dijalankan pada piranti komputer yang akan disebut arsitektur. Arsitektur yang dimaksud dapat berupa sebuah komputer atau mungkin juga sebuah perangkat keras yang berfungsi secara spesifik untuk tugas tertentu seperti kamera pemroses image atau filter input audio. Secara umum, arsitektur membuat *percept* dari sensor kemudian siap digunakan ke dalam program, menjalankan program, dan memberikan pilihan-pilihan tindakan program ke efektor seperti yang mereka hasilkan.

Dalam merancang agen yang baik harus memperhatikan 4 hal yaitu *Percept, Action, Goal,* dan *Environment*; biasa disingkat PAGE (Russell and Norvig, 1995). Agen harus memahami rangsangan (*percept*) yang diterima melalui sensor, kemudian melakukan tindakan sebagai respon dari pemahaman yang diterima dan tindakan (*action*) yang dilakukan oleh agen harus disesuaikan dengan tujuan (*goal*) yang hendak dicapai. Semua tindakan (*environment*) yang dilakukan oleh agen harus memperhatikan di lingkungan mana agen tersebut berada.

2.2. Tipe Agen

Terdapat 4 tipe Agen :

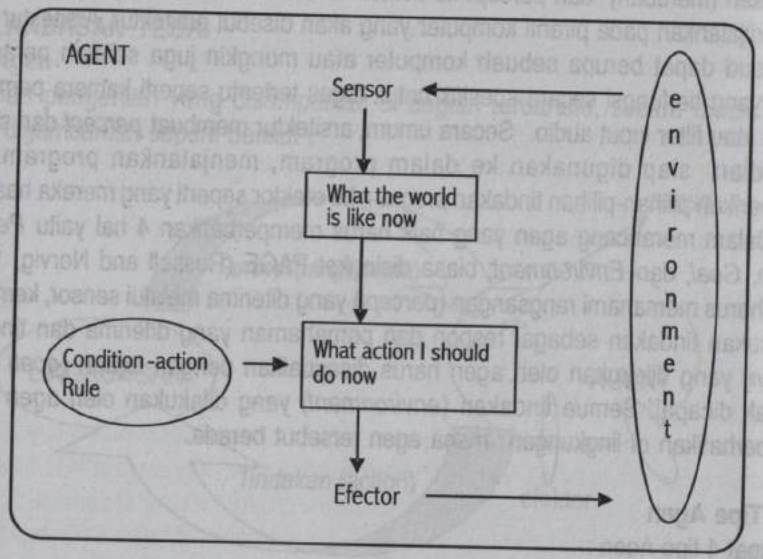
1. *Simple reflex Agen*
2. *Agent that keep track of the world*

3. *Goal-based Agent (*
4. *Utility-based Agent*

Simple Reflex Agent adalah agen yang bekerja berdasarkan reflek. Hal ini juga seperti pada manusia yang memberikan respon karena reflek. Dalam tipe agen ini beberapa proses dikerjakan berdasarkan input visual untuk menghasilkan kondisi yang baik. Gambar 2 berikut adalah struktur *simple reflex agent*, menunjukkan bagaimana *condition-action rule* (biasa disebut juga *if-then rule*) mengijinkan agen untuk membuat hubungan dari *percept* ke *action*. Notasi yang digunakan dalam struktur adalah :

: internal state proses keputusan agen

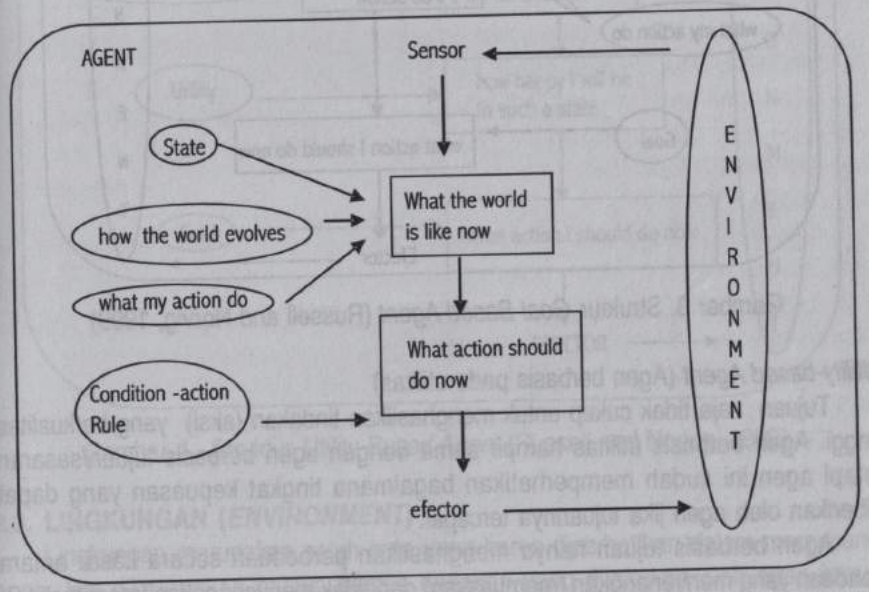
: latar belakang informasi yang digunakan dalam proses.



Gambar 2. Struktur *Simple Reflex Agent* (Russell and Norvig, 1999)

Agent that keep track of the world

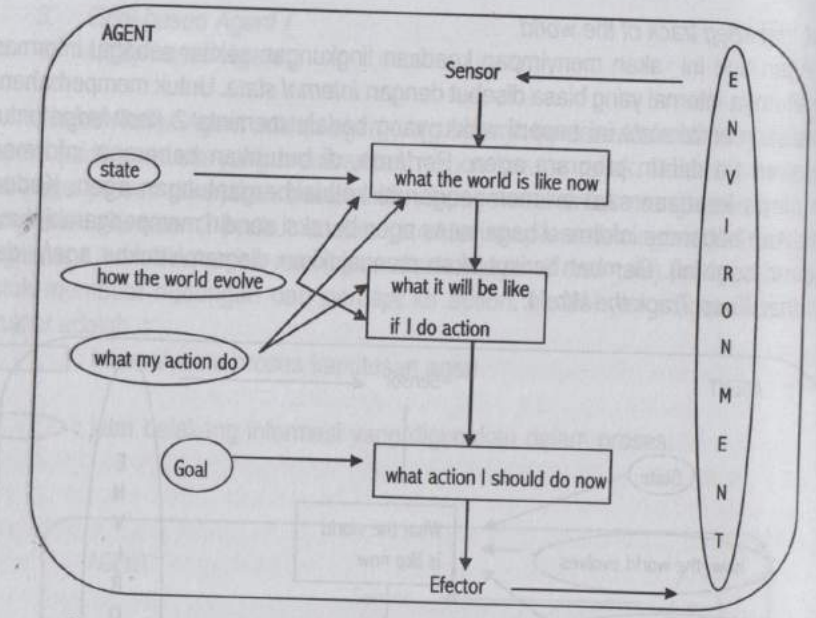
Agen tipe ini akan menyimpan keadaan lingkungan sekitar sebagai informasi yang sifatnya internal yang biasa disebut dengan *internal state*. Untuk memperbaharui informasi *internal state* ini seperti waktu yang berlalu meminta 2 *knowledge* untuk dikodekan ke dalam program agen. Pertama, di butuhkan beberapa informasi bagaimana keadaan saat ini mempengaruhi ketidakbergantungan agen. Kedua, dibutuhkan beberapa informasi bagaimana agen beraksi sendiri mempengaruhi dunia (keadaan saat ini). Gambar berikut akan menunjukkan diagram/struktur agen dari *agen that Keep Track the World:*



Gambar 3. Struktur *Agent that Keep Track the World* (Russell and Norvig, 1999)

Goal-based Agent (Agen berbasis pada tujuan)

Agen berdasarkan/berbasis tujuan merupakan sebuah agen yang semua tindakannya didasarkan pada tujuan yang akan dicapai. Tindakan/aksi yang dilakukan oleh agen harus mempertimbangkan keadaan yang akan terjadi (keadaan yang akan datang), misalnya jika agen melakukan aksi A apa yang akan terjadi setelah melakukan tindakan tersebut. Gambar struktur agen tipe ini dapat dilihat pada gambar 3.

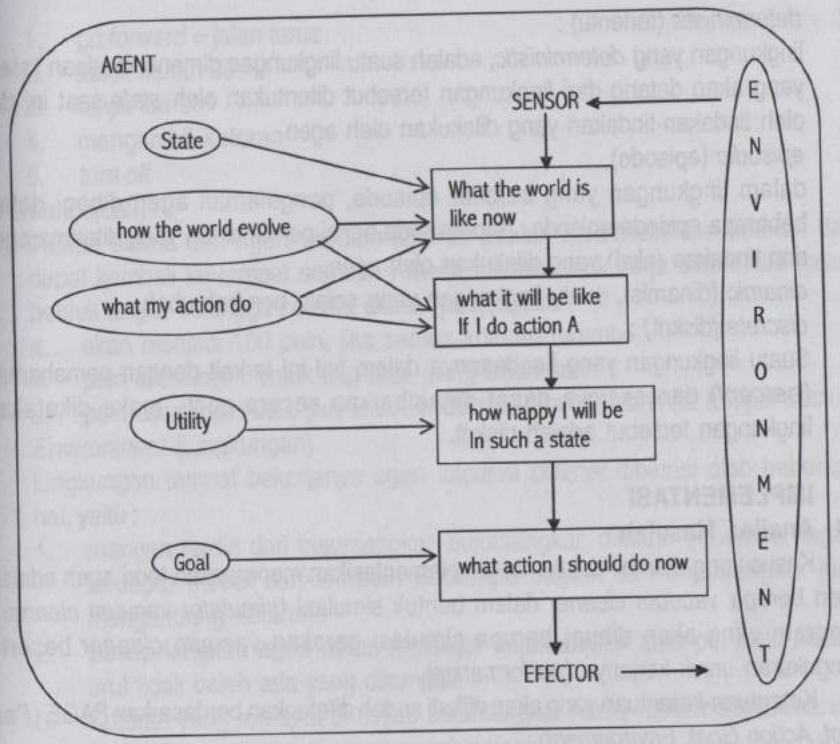


Gambar 3. Struktur *Goal Based Agent* (Russell and Norvig, 1999)

Utility-based Agent (Agen berbasis pada utilitas)

Tujuan saja tidak cukup untuk menghasilkan tindakan (aksi) yang berkualitas tinggi. Agen berbasis utilitas hampir sama dengan agen berbasis tujuan/sasaran tetapi agen ini sudah memperhatikan bagaimana tingkat kepuasan yang dapat diberikan oleh agen jika tujuannya tercapai.

Agen berbasis tujuan hanya menghasilkan perbedaan secara kasar antara keadaan yang menyenangkan (memuaskan) dan tidak menyenangkan (memuaskan); dimana ukuran kinerja secara umum seharusnya memberikan perbandingan untuk keadaan (atau urutan keadaan) yang berbeda ditambahkan secara jelas bagaimana tingkat kepuasan yang diberikan oleh agen jika tujuannya tercapai. Gambar 4 menggambarkan struktur agen berbasis pada utilitas.



Gambar 4. Struktur *Utility Based Agent* (Russell and Norvig, 1999)

2.3. LINGKUNGAN (*ENVIRONMENT*)

Lingkungan merupakan salah satu yang harus diperhatikan dalam merancang agen. Hal dikarenakan bahwa semua tindakan yang dilakukan oleh agen harus memperhatikan/mempertimbangkan kondisi/keadaan (biasa disebut lingkungan) sekitar di aman agen tersebut berada/bereaksi. Terdapat beberapa kriteria pembagian lingkungan :

1. *accessible* (dapat diakses) :
   suatu lingkungan dikatakan *accessible*, jika sensor agen dapat mendeteksi semua keadaan limgkungan, terutama yang berkaitan dengan pilihan tindakan (aksi) yang harus dilakukan.

2. *deterministic* (tertentu) :
   lingkungan yang *deterministic*, adalah suatu lingkungan dimana keadaan (*state*) yang akan datang dari lingkungan tersebut ditentukan oleh *state* saat ini dan oleh tindakan-tindakan yang dilakukan oleh agen.
3. *episodic* (episode),
   dalam lingkungan yang bersifat episode, pengalaman agen dibagi dalam beberapa episode-episode. Tiap episode berisi pemahaman yang diterima agen dan tindakan (aksi) yang dilakukan oleh agen.
4. *dinamic* (dinamis), suatu lingkungan yang selalu berubah-ubah.
5. *discrete* (diskrit) :
   Suatu lingkungan yang keadaannya dalam hal ini terkait dengan pemahaman (*percept*) dan aksinya dapat digambarkan secara pasti, maka dikatakan lingkungan tersebut adalah diskrit.

3. IMPLEMENTASI

3.1. Analisa Masalah

Kasus yang diambil untuk mengimplementasikan/menerapkan teori agen adalah agen berupa *vacuum cleaner* dalam bentuk simulasi (*simulator vacuum cleaner*). Program yang akan dibuat berupa simulasi gerakan *vacuum cleaner* beserta pengukuran unjuk kerjanya (*performance*).

Ketentuan-ketentuan yang akan diikuti sudah ditetapkan berdasarkan PAGE (*Percept, Action Goal, Environment*) :

a. *Percept* (persepsi) :
   Setiap agen *vacuum cleaner* memberikan 3 elemen persepsi :
   1. sensor sentuhan
      menunjuk 1 jika menyentuh sesuatu, menunjuk 0 jika sebaliknya
   2. sensor foto
      menunjuk 1 jika terdapat kotoran, menunjuk 0 jika sebaliknya
   3. sensor infra red
      menunjuk 1 jika agent kembali ke tempat semula, menunjuk 0 jika tidak kembali

b. *Actions* (tindakan, aksi)
   Terdapat 5 aksi :

   1. *go forward* = jalan terus
   2. belok kanan 90 ^{0}
   3. belok kiri 90^0
   4. mengambil kotoran
   5. *turn off*

c. *Goal* (tujuan)
   Tujuan utama dari agen *vacuum cleaner* adalah membersihkan kotoran dan dapat kembali ke tempat semula. Kriteria ukuran unjuk kerja diberi nilai dalam bentuk angka sehingga bersifat eksak (pasti), yaitu :
   a. akan menjadi 100 poin, jika semua kotoran terambil
   b. poin dikurangi 1 untuk tiap aksi yang dillakukan
   c. poin dikurangi 1000, jika *vacuum cleaner* tidak kembali ke tempat semula

d. *Environment* (Lingkungan)
   Lingkungan tempat bekerjanya agen *vacuum cleaner* dibatasi oleh beberapa hal, yaitu :
   1. ruangan terdiri dari bujursangkar–bujursangkar, dimana beberapa bagian terdapat mebel dan tembok. Beberapa bagian dari bujursangkar juga mengandung kotoran.
   2. Setiap langkah agen harus melewati bujursangkar satu per satu secara urut tidak boleh ada yang dilompati.
   3. Kotoran yang terdapat di setiap bujursangkar harus selalu dibersihkan.
   4. *Turn off* terjadi, jika simulasi sudah selesai

Dalam agen ini juga dibatasi bahwa mebel dan tembok tidak bisa dibedakan sensor sentuhan sehingga keduanya tetap bernilai 1. Sampah juga didistribusikan secara tidak beraturan di sekeliling ruang.

3.2. Perancangan

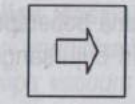
Asumsi-asumsi terkait dengan PAGE yang akan digunakan dalam pembuatan program adalah sebagai berikut :

1. Bentuk ruang n x n
2. Peletakan awal *vacuum cleaner* selalu pada posisi pojok kiri atas (baris 1, kolom1)
3. Peletakan sampah dan mebel bebas
4. Simbol-simbol yang digunakan :

1. 'X' : sampah
2. 'M' : mebel
3. '.' : area bebas sampah dan mebel
4. '>' : *vacuum* bergerak ke kanan layar
5. '<' : *vacuum* bergerak ke kiri layar
6. 'v' : *vacuum* bergerak ke bawah layar
7. '^' : *vacuum* bergerak ke atas layar
8. '0' dan '*' : proses *cleaning* (pengambilan sampah)

5. Prioritas gerakan *vacuum cleaner* :

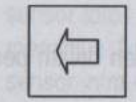
   a. *vacuum cleaner* bergerak ke kanan jika bertemu tembok atau mebel, maka prioritas gerakan berikutnya adalah ke bawah (belok kanan 90^0) baru kemudian ke atas (belok kiri 90^0). Bila keduanya tidak memungkinkan (posisi terjepit), maka *vacuum cleaner* hanya akan hadap kanan (belok kanan 90^0 masih dalam grid yang sama)



posisi terjepit

hadap kanan

   b. *vacuum cleaner* bergerak ke kiri jika bertemu tembok atau mebel, maka prioritas gerakan berikutnya adalah ke bawah (belok kiri 90^0) baru kemudian ke atas (belok kanan 90^0). Bila keduanya tidak memungkinkan (posisi terjepit), maka *vacuum cleaner* hanya akan hadap kiri (belok kiri 90^0 masih dalam grid yang sama)

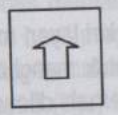


posisi terjepit

hadap kiri

   c. *vacuum cleaner* bergerak ke atas jika bertemu tembok atau mebel, maka prioritas gerakan berikutnya adalah ke kanan layar (belok kanan 90^0) baru kemudian ke kiri layar (belok kiri 90^0). Bila keduanya tidak memungkinkan

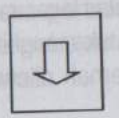
(posisi terjepit), maka *vacuum cleaner* hanya akan hadap kanan (belok kanan 90^0 masih dalam grid yang sama).



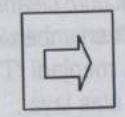
posisi terjepit

hadap kanan

   d. *vacuum cleaner* bergerak ke bawah jika bertemu tembok atau mebel, maka prioritas gerakan berikutnya adalah ke kanan layar (belok kiri 90^0) baru kemudian ke kiri layar (belok kanan 90^0) Bila keduanya tidak memungkinkan (posisi terjepit), maka vacuum cleaner hanya akan hadap kiri (belok kiri 90^0 masih dalam grid yang sama)



posisi terjepit



hadap kiri

Penyelesaian masalah di atas adalah dengan membuat suatu program yang mengikuti aturan-aturan yang sudah ditetapkan dan mengacu dari asumsi-asumsi yang ada. Program terdiri beberapa sub rutin yang mengimplementasikan aturan yang telah ditetapkan dan asumsi yang telah dibuat. Secara keseluruhan program dibuat untuk dapat memvisualisasikan gerakan *vacuum cleaner* dan mengukur unjuk kerjanya.

Rancangan subrutin yang harus dibuat adalah sebagai berikut :

1. Sub rutin untuk membuat ruang,
   fungsinya untuk menentukan besarnya sisi dari ruangan dan penentuan isinya (peletakan sampah dan mebel) serta penentuan status apakah grid pernah dilewati atau belum oleh *vacuum cleaner*.
2. Sub rutin untuk menampilkan visualisasi gerakan *vacuum cleaner*.
3. Sub rutin untuk bergerak ke kanan,
   fungsinya untuk menggerakkan *vacuum cleaner* ke arah kanan layar, menampilkan simbol gerakan *vacuum cleaner*, menambah *counter* (untuk

menghitung jumlah aksi), dan mengganti status grid menjadi 'R' artinya pernah dilewati *vacuum cleaner* dengan gerakan ke arah kanan layar.

4. Sub rutin untuk bergeser ke kiri,
   fungsinya untuk menggerakkan *vacuum cleaner* ke arah kiri layar, menampilkan simbol gerakan *vacuum cleaner*, menambah *counter* (untuk menghitung jumlah aksi), dan mengganti status grid menjadi 'L' artinya pernah dilewati *vacuum cleaner* dengan gerakan ke arah kiri layar.
5. Sub rutin untuk bergeser ke bawah,
   fungsinya untuk menggerakkan *vacuum cleaner* ke arah bawah layar, menampilkan simbol gerakan *vacuum cleaner*, menambah *counter* (untuk menghitung jumlah aksi), dan mengganti status grid menjadi 'B' artinya pernah dilewati *vacuum cleaner* dengan gerakan ke arah bawah layar.
6. Sub rutin untuk bergeser ke atas,
   fungsinya untuk menggerakkan *vacuum cleaner* ke arah atas layar, menampilkan simbol gerakan *vacuum cleaner*, menambah *counter* (untuk menghitung jumlah aksi), dan mengganti status grid menjadi 'T' artinya pernah dilewati *vacuum cleaner* dengan gerakan ke arah atas layar.
7. Sub rutin untuk hadap kanan,
   fungsinya untuk menggerakkan *vacuum cleaner* menghadap kanan layar, menampilkan simbol gerakan *vacuum cleaner*, menambah *counter* (untuk menghitung jumlah aksi).
8. Subrutin untuk hadap ke bawah,
   fungsinya untuk menggerakkan *vacuum cleaner* menghadap bawah layar, menampilkan simbol gerakan *vacuum cleaner*, menambah *counter* (untuk menghitung jumlah aksi).
9. Sub rutin untuk menampilkan simbol gerakan proses pengambilan sampah oleh *vacuum cleaner*, menambah *counter* (untuk menghitung jumlah aksi), dan mengganti simbol grid menjadi ' . ' artinya sampah sudah diambil dan area menjadi bersih.
10. Sub rutin untuk menerapkan dari prioritas gerakan *vacuum cleaner* (sesuai dengan penjelasan tentang prioritas di atas)
11. Sub rutin utnuk menampilkan hasil kerja dan unjuk kerja *vacuum cleaner.*

3.3. Pembahasan

Pembahasan ini akan menampilkan hasil dari program yang telah dibuat, dimana pemakai bebas memasukkan ukuran lantai dan peletakan mebel maupun sampah. Pembahasan hasil program ini dimulai tidak dari saat pemasukan data tetapi sudah dala tampilan bentuk simulasinya (proses input data tidak disertakan)

Terdapat 3 (tiga) kriteria kasus yang dapat dibahas dalam program ini :

1. kasus dimana vacuum *cleaner* dapat menyelesaikan pekerjaan (dapat mengambil semua sampah dan kembali ke tempat semula).
2. kasus dimana *vacuum cleaner* berputar menempuh rute yang sama, tetapi dapat kembali ke tempat semula (sampah tidak dapat diambil semua).
3. kasus dimana *vacuum cleaner* berputar menempuh rute yang sama dan tidak dapat kembali ke tempat semula (sampah tidak dapat diambil semua).

Contoh kasus pertama :

Kondisi awal sebelum *action* dilakukan, hasil penentuan letak sampah dan mebel sebagai berikut :

```
Ruang 6 x 6
-----------
>  .  X  .  X  M
.  .  .  .  .  .
.  M  .  .  .  .
.  X  .  .  .  .
.  .  X  .  M  .
X  M  .  X  .  X

Action : 0
Tekan sembarang tombol

<enter>
```

Kondisi akhir setelah *action* dilakukan :

```
Ruang 6 x 6
-------
^  .  .  .  .  M
.  .  .  .  .  .
.  M  .  .  .  .
.  .  .  .  .  .
.  .  .  .  M  .
.  M  .  .  .  .

Action : 58 jalan maju
Vacuum menyelesaikan pekerjaan
Jumlah sampah   : 7
Sampah diambil  : 7
Jumlah action   : 58

Skor akhir      : 42.00
```

Pada kasus pertama ini *vacuum cleaner* berhasil menyelesaikan pekerjaan sepenuhnya, yaitu mengambil semua sampah yang ada dan kembali ke posisi semula (pojok kiri atas). Jumlah *action* yang dilakukan sebanyak 58 aksi, diperoleh skor akhir sebagai ukuran performance kerja *vacuum cleaner* = 100 – 58 = 42.

Contoh kasus kedua :

Kondisi awal sebelum *action* dilakukan, hasil penentuan letak sampah dan mebel sebagai berikut :

```
Ruang 6 x 6
-------
>  .  X  .  X  .
.  .  .  .  .  M
.  M  .  .  .  .
.  X  .  .  .  .
.  .  X  .  M  .
X  M  .  X  .  X

Action : 0
Tekan sembarang tombol

<enter>
```


Kondisi akhir setelah *action* dilakukan :

```
Ruang 6 x 6
-------
^  .  .  .  .  .
.  .  .  .  .  M
.  M  .  .  .  .
.  X  .  .  .  .
.  .  X  .  M  .
.  M  .  X  .  X

Action : 47 belok kanan
Rute pernah dilalui
Vacuum menghentikan pekerjaan
Jumlah sampah   : 7
Sampah diambil  : 3
Jumlah action   : 47

Skor akhir      : -4.14
```

Pada kasus kedua ini *vacuum cleaner* tidak berhasil menyelesaikan pekerjaan sepenuhnya, karena misalnya *vacuum cleaner* ketemu jalan buntu sehingga dengan *action* yang ada tidak dapat melanjutkan proses hanya berputar-putar saja, tetapi karena perputaran itu melewati posisi awal *vacuum cleaner* sehingga *vacuum cleaner* dapat kembali ke tempat semula meskipun sampah tidak dapat terambil semua. Dalam program diberikan suatu kondisi, *vacuum cleaner* melewati posisi awal (kolom 1 baris 1) maksimal sebanyak 4 kali, selebihnya *vacuum cleaner* akan menghentikan proses dan kembali ke posisi awal. Contoh di atas *vacuum cleaner* hanya mampu mengambil 3 sampah dari keseluruhan sampah dan jumlah aksi sebanyak 47, sehingga skor ukuran performance = (3/7 x 100) – 47 = - 4,14.

Contoh kasus ketiga

Kondisi awal sebelum *action* dilakukan, hasil penentuan letak sampah dan mebel sebagai berikut :

```
Ruang 6 x 6
-----------
>  .  X  .  M  .
.  .  .  .  X  .
.  M  .  .  .  .
.  X  .  .  .  .
M  .  X  .  M  .
X  .  .  X  .  X

Action : 0
Tekan sembarang tombol
<enter>
```

Kondisi awal setelah *action* dilakukan :

```
Ruang 6 x 6
-----------
.  .  .  .  M  .
.  .  .  .  X  .
.  M  .  .  .  .
.  X  .  .  .  .
M  .  X  .  M  .
.  .  .  <  .  .

Action : 181 jalan maju
Rute selalu berputar
Vacuum berhenti di tengah jalan

Jumlah sampah  : 7
Sampah diambil : 4
Jumlah action  : 181

Skor akhir     : -1123.86
```

Pada kasus ketiga ini *vacuum cleaner* tidak berhasil menyelesaikan pekerjaan sepenuhnya, misalnya *vacuum cleaner* ketemu kondisi di mana rutenya hanya berputar-putar saja dan *vacuum cleaner* tidak pernah melewati posisi awal sehingga tidak dapat kembali ke posisi semula. Kasus ketiga ini memungkinkan adanya sampah yang tidak dapat terambil semua. Pada implementasi program dibatasi aksi maksimal

adalah 5 x n x n, jika *action* lebih dari 5 x n x n maka *vacuum cleaner* akan menghentikan pekerjaan dimanapun posisinya. Hal tersebut menyebabkan adanya kemungkinan sampah tidak terambil semua. Dari contoh di atas, 7 sampah hanya bisa terambil 4, jumlah *action* yang dilakukan 181 dan *vacuum* tidak dapat kembali ke posisi semula (nilai di kurangi 1000),sehingga skor untuk *performance-* nya dapat dihitung sebagai berikut = (4/7 x 100)-181–1000 = -1123,86.

4. KESIMPULAN DAN SARAN

4.1. Kesimpulan

1. Agen yang berhasil dibuat masih berupa simulasi dalam bentuk sederhana.
2. Implementasi permasalahan "*vacuum cleaner*" dalam bentuk program agen, meskipun sudah diberikan tambahan asumsi tetap saja ada keterbatasan yaitu adanya kasus dimana suatu saat sampah tidak dapat terambil semua dan *vacuum cleaner* tidak dapat kembali ke tempat semula, sehingga tujuan (*goal*) tidak dapat tercapai 100 %, berakibat unjuk kerja *vacuum cleaner* tidak maksimal.

4.2. Saran

1. Program Agen yang berhasil dibuat perlu ditambahkan (dimunculkan) secara eksplisit indikator yang menunjukkan 3 elemen persepsi yang diterima oleh agen (status *touch sensor*, *photo sensor*, dan *infrared sensor*).
2. Program agen yang dibuat perlu dikembangkan tidak hanya dalam bentuk simulasi, tetapi mengarah pada bentuk yang sesungguhnya dengan memperhatikan arsitekturnya.

5. DAFTAR PUSTAKA

Kadir, Abdul, dan Triwahyuni, Ch., T., 2003, "Pengenalan Teknologi Informasi", Penerbit ANDI, Yogyakarta

Russell J.,S., and Norvig P., 1995, "Artificial Intelligence : a Modern Approach", Prentice-Hall inc., USA

Wahono, S.,R., "Pengantar Software Agent: Teori dan Aplikasi", http://IlmuKomputer.Com