TUGAS AKHIR – KI141502

IMPLEMENTASI ALGORITMA DIJKSTRA PADA PERMAINAN TATTARA SEBAGAI PEMBANGKIT *PUZZLE* DAN PEMBERI SKOR

MUHAMMAD HUSAIN FUAD DZULFIKRI
NRP. 5112100046

Dosen Pembimbing 1
Imam Kuswardayan, S.Kom., M.T.

Dosen Pembimbing 2
Anny Yuniarti, S.Kom.,M.Comp.Sc

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informatika
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – KI141502

IMPLEMENTASI ALGORITMA DIJKSTRA PADA PERMAINAN TATTARA SEBAGAI PEMBANGKIT *PUZZLE* DAN PEMBERI SKOR

MUHAMMAD HUSAIN FUAD DZULFIKRI
NRP. 5112100046

Dosen Pembimbing 1
Imam Kuswardayan, S.Kom., M.T.

Dosen Pembimbing 2
Anny Yuniarti, S.Kom.,M.Comp.Sc

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informatika
Institut Teknologi Sepuluh Nopember
Surabaya 2018

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – KI141502

DIJKSTRA ALGORITHM IMPLEMENTATION IN TATTARA GAME AS PUZZLE AND SCORE MAKER

MUHAMMAD HUSAIN FUAD DZULFIKRI
NRP. 5112100046

Supervisor 1
Imam Kuswardayan, S.Kom., M.T.

Supervisor 2
Anny Yuniarti, S.Kom.,M.Comp.Sc

DEPARTMENT OF INFORMATICS
Faculty of Informatics Technology
Sepuluh Nopember Institute of Technology
Surabaya 2018

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

IMPLEMENTASI ALGORITMA DIJKSTRA PADA PERMAINAN TATTARA SEBAGAI PEMBANGKIT *PUZZLE* DAN PEMBERI SKOR

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Grafis dan Seni
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
MUHAMMAD HUSAIN FUAD DZULFIKRI
NRP. 5112100046

Disetujui oleh Pembimbing Tugas Akhir:

1. Imam Kuswardayan, S.Kom., M.T. .....
   NIP. 197612152003121001 (Pembimbing 1)

2. Anny Yuniarti, S.Kom.,M.Comp.Sc .....
   NIP. 198106222005012002 (Pembimbing 2)

SURABAYA
JULI 2018

vii

[Halaman ini sengaja dikosongkan]

IMPLEMENTASI ALGORITMA DIJKSTRA PADA PERMAINAN TATTARA SEBAGAI PEMBANGKIT *PUZZLE* DAN PEMBERI SKOR

Nama Mahasiswa	**: Muhammad Husain Fuad Dzulfikri**
NRP	**: 5112100046**
Jurusan	**: Teknik Informatika Fakultas Teknologi Informasi ITS**
Pembimbing I	**: Imam Kuswardayan, S.Kom., M.T.**
Pembimbing II	**: Anny Yuniarti, S.Kom.,M.Comp.Sc**

ABSTRAK

Pembuatan permainan *puzzle* agar saat dimainkan nantinya tidak membosankan berulang kali merupakan salah satu alasan mengapa saat melakukan pembuatan permainan perlu memerhatikan tingkat *playability* permainan yang dibuat. Selain agar tidak membosankan, semakin tinggi tingkat *playability* suatu permainan maka memiliki dampak meningkatnya nilai permainan tersebut. Jenis permainan *puzzle* cukup mudah dimainkan serta memiliki cara permainan yang cukup sederhana. Permainan *puzzle* akan sering terus diulang namun dengan tingkatan berbeda yang memberikan rasa permainan yang menarik, sehingga memiliki tingkat *playable* yang tinggi.

Untuk mendapatkan permainan *puzzle* yang memiliki tingkat *playability* tinggi, maka dalam permainan *puzzle* dapat diimplementasikan pembangkitan *puzzle* permainan yang menarik pada permainan. Permainan juga memiliki pembangkitan yang acak antar gambar *puzzle* yang perlu adanya metode untuk memastikan pengacakan yang telah dilakukan dapat diselesaikan. Sehingga setiap kali permainan tidak menemukan solusi maka permainan *puzzle* yang baru dapat dibangkitkan. Sehingga menciptakan meningkatnya keanekaragaman permainan *puzzle* yang ada di Indonesia. Juga menjadi sarana inspirasi untuk

pengembang permainan lainnya dalam membuat permainan yang menarik.

Kata kunci: permainan, *playability, puzzle.*

DIJKSTRA ALGORITHM IMPLEMENTATION IN TATTARA GAME AS PUZZLE AND SCORE MAKER

Name	**: Muhammad Husain Fuad D**
NRP	**: 5112100046**
Department	**: Department of Informatics Faculty of Information Technology ITS**
Supervisor I	**: Imam Kuswardayan, S.Kom., M.T.**
Supervisor II	**: Anny Yuniarti, S.Kom.,M.Comp.Sc**

ABSTRACT

Making a puzzle game so that when played later is not boring repeatedly is one of the reasons why when making a game need to pay attention to the level of playability of the game is made. In *add*ition to not boring, the higher the playability level of a game then it has the impact of increasing the value of the game. This type of puzzle game is quite easy to play and has a fairly simple game. The puzzle game will often be repeated many times but with different levels that give the game an interesting feel, so it has a high playable level.

To get a puzzle game that has a high playability level, then in the puzzle game can be implemented an interesting game puzzle generation on the game. The game also has a random generation of puzzle images that need a method to ensure that randomization has been completed. So every time the game does not find a solution then a new puzzle game can be raised. Thus creating an increasing diversity of puzzle games in Indonesia. It is also a means of inspiration for other game developers in creating exciting games

Keywords : game, playability, puzzle.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Alhamdulillahirabbil'alamiin, puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena atas segala karunia dan rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul **"Implementasi Algoritma Dijkstra pada Permainan Tattara Sebagai Pembangkit *Puzzle* dan Pemberi Skor".**

Tugas akhir ini dilakukan untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer di Jurusan Teknik Informatika Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember.

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan baik secara langsung maupun tidak langsung selama proses pengerjaan tugas akhir ini hingga selesai, antara lain:

1. Tuhan Yang Maha Esa atas segala karunia dan rahmat-Nya yang telah diberikan selama ini.
2. Orang tua, saudara serta keluarga penulis yang tiada henti-hentinya memberikan semangat, perhatian dan doa selama perkuliahan penulis di Jurusan Teknik Informatika ini.
3. Bapak Imam Kuswardayan, S.Kom., M.T. selaku dosen pembimbing I yang telah memberikan bimbingan, arahan dan motivasi dalam pengerjaan tugas akhir ini.
4. Ibu Anny Yuniarti, S.Kom.,M.Comp.Sc selaku dosen pembimbing II yang telah banyak memberikan arahan dan bantuan dalam pengerjaan tugas akhir ini.
5. Bapak Radityo Anggoro, S.Kom., M.Sc., Dr.Eng., selaku kepala program pendidikan Teknik Informatika yang telah banyak memberikan arahan dan bantuan, waktu, dan motivasi sehingga penulis dapat menyelesaikan tugas akhir ini.

6. Seluruh Bapak dan Ibu dosen Jurusan Teknik Informatika yang telah memberikan banyak ilmu dan pengalamannya.
7. Seluruh staf dan karyawan Jurusan Teknik Informatika yang banyak memberikan kelancaran administrasi akademik kepada penulis.
8. Teman-teman penulis yang telah membantu penulis menyelesaikan tugas akhir ini.
9. Seluruh pihak yang tidak dapat penulis sebutkan satu persatu yang telah memberikan dukungan selama penulis menyelesaikan tugas akhir ini.

Penulis mohon maaf apabila terdapat kekurangan dalam penulisan buku tugas akhir ini. Kritik dan saran penulis harapkan untuk perbaikan dan pembelajaran di kemudian hari. Semoga tugas akhir ini dapat memberikan manfaat yang sebaik-baiknya.

Surabaya, Juli 2018

Muhammad Husain Fuad Dzulfikri

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini akan dijelaskan hal-hal yang menjadi latar belakang, permasalahan yang dihadapi, batasan masalah, tujuan, metodologi dan sistematika penulisan yang digunakan dalam pembuatan buku tugas akhir ini.

1.1. Latar Belakang

Pembuatan permainan *puzzle* agar saat dimainkan nantinya tidak membosankan berulang kali merupakan salah satu alasan mengapa saat melakukan pembuatan permainan perlu memerhatikan tingkat *playability* permainan yang dibuat. Selain agar tidak membosankan, semakin tinggi tingkat *playability* suatu permainan maka memiliki dampak meningkatnya nilai permainan tersebut. Jenis permainan *puzzle* cukup mudah dimainkan serta memiliki cara permainan yang cukup sederhana. Permainan *puzzle* akan sering terus diulang namun dengan tingkatan berbeda yang memberikan rasa permainan yang menarik, sehingga memiliki tingkat *playable* yang tinggi.

Untuk mendapatkan permainan *puzzle* yang memiliki tingkat *playability* tinggi, maka dalam permainan *puzzle* dapat diimplementasikan pembangkitan *puzzle* permainan yang menarik pada permainan. Permainan juga memiliki pembangkitan yang acak antar gambar *puzzle* yang perlu adanya metode untuk memastikan pengacakan yang telah dilakukan dapat diselesaikan. Sehingga setiap kali permainan tidak menemukan solusi maka permainan *puzzle* yang baru dapat dibangkitkan.

Dari permainan ini diharapkan akan meningkatnya keanekaragaman permainan *puzzle* yang ada di Indonesia. Juga

menjadi sarana inspirasi untuk pengembang permainan lainnya dalam membuat permainan yang menarik.

1.2. Rumusan Permasalahan

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana penentuan aturan main, skenario, dan level pada permainan Tattara?
2. Bagaimana *puzzle* yang dibangkitkan secara acak dapat diselesaikan?
3. Bagaimana menentukan aturan pemberian skor permainan berdasarkan *puzzle* pilihan pemain?

1.3. Tujuan dan Manfaat

Tujuan dari pembuatan tugas akhir ini adalah membuat aplikasi permainan *puzzle* yang mengimplementasikan pembangkitan skor berdasarkan rute yang ditempuh potongan *puzzle*. Pemastian pemetaan rute-rute potongan dapat diselesaikan untuk menghindari kebuntuan di dalam permainan sebagai sarana meningkatkan tingkat *playability* dan meningkatkan nilai permainan tersebut.

Manfaat dari pembuatan tugas akhir ini adalah:

1. Mengimplementasikan pembangkitan *puzzle* secara acak yang selalu dapat diselesaikan serta pencarian rute terpanjang yang dapat dilalui untuk menghubungkan potongan *puzzle*.
2. Mengimplementasikan pemberian skor yang berbeda berdasarkan pilihan jarak *puzzle*.
3. Meningkatkan kekreatifan dalam menyiapkan jenis permainan *puzzle* yang berbeda.
4. Memberikan media hiburan bagi para pengguna.

1.4. Batasan Permasalahan

Tujuan dari pembuatan tugas akhir ini adalah membuat aplikasi permainan *puzzle* yang mengimplementasikan pembangkitan skor berdasarkan rute yang ditempuh potongan *puzzle*. Pemastian pemetaan rute-rute potongan dapat diselesaikan untuk menghindari kebuntuan di dalam permainan sebagai sarana meningkatkan tingkat *playability* dan meningkatkan nilai permainan tersebut.

Manfaat dari pembuatan tugas akhir ini adalah:

1. Mengimplementasikan pembangkitan *puzzle* secara acak yang selalu dapat diselesaikan serta pencarian rute terpanjang yang dapat dilalui untuk menghubungkan potongan *puzzle*.
2. Mengimplementasikan pemberian skor yang berbeda berdasarkan pilihan jarak *puzzle*.
3. Meningkatkan kekreatifan dalam menyiapkan jenis permainan *puzzle* yang berbeda.
4. Memberikan media hiburan bagi para pengguna.

1.5. Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Studi literatur

   Pada studi literatur ini, akan dipelajari sejumlah referensi yang diperlukan dalam pembuatan permainan *puzzle* serta referensi pembangkitan skor permainan *puzzle* dari pemetaan rute-rute secara acak.
2. Analisis dan perancangan sistem

Pada tahap ini dilakukan analisa dan mendefinisikan fitur pada aplikasi. Dari proses tersebut selanjutnya dirumuskan rancangan yang akan digunakan. Langkah-langkah pada tahap ini adalah sebagai berikut :

a. Perancangan diagram *use-case*, yang merupakan analisis kebutuhan pada aplikasi yang akan dibangun
b. Fitur yang akan terdapat pada aplikasi ini diantaranya adalah:
    1. Jumlah pemain yaitu satu orang.
    2. Grafik 2D.
    3. Jenis permainan adalah *puzzle*.

3. Pembuatan (implementasi)

Pada tahap ini dilakukan pembuatan perangkat lunak yang merupakan implementasi dari rancangan yang telah dibuat sebelumnya.

Perincian tahap ini adalah sebagai berikut:

a. Implementasi antarmuka aplikasi
b. Implementasi algoritma untuk menentukan solusi

4. Uji coba dan evaluasi

Pengujian permainan ini akan dilakukan dengan pengujian *playable*. Pengujian ini dilakukan untuk menguji apakah pembangkitan permainan *puzzle* acak di dalam permainan sudah diimplementasikan dan pemberian skor permainan bisa didapatkan dari rute pencarian pasangan potongan *puzzle*.

5. Penyusunan buku tugas akhir

Pada tahap ini melakukan pendokumentasian dan laporan dari seluruh konsep, dasar teori, implementasi, proses yang telah dilakukan, dan hasil-hasil yang telah didapatkan selama pengerjaan tugas akhir. Buku tugas

akhir ini bertujuan untuk memberikan gambaran dari pengerjaan tugas akhir ini dan diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut.

1.6. Sistematika Penulisan

Pendokumentasian dan laporan dari seluruh konsep, dasar teori, implementasi, proses yang telah dilakukan, dan hasil-hasil yang telah didapatkan selama pengerjaan tugas akhir. Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini dan diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut.

Secara garis besar, buku tugas akhir nantinya terdiri atas beberapa bagian yaitu :

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

Bab II Kajian Pustaka

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan

Bab ini membahas mengenai desain dan perancangan perangkat lunak. Desain perangkat lunak meliputi desain data, arsitektur, dan proses.

Bab IV Implementasi

Bab ini membahas implementasi dari rancangan sistem yang dilakukan pada tahap perancangan.

Bab V Pengujian dan Evaluasi

Bab ini membahas pengujian dari aplikasi yang dibuat dengan melihat *output* yang dihasilkan oleh aplikasi, dan evaluasi untuk mengetahui kemampuan aplikasi.

Bab VI Kesimpulan dan Saran

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan serta saran-saran untuk pengembangan aplikasi pada masa mendatang.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

BAB II
KAJIAN PUSTAKA

Bab ini akan membahas mengenai dasar teori dan literatur yang menjadi dasar pengerjaan tugas akhir ini.

2.1. Algoritma Dijkstra

Algoritma ini dinamai berdasarkan nama penemunya, seorang ilmuwan komputer bernama Edsger Dijkstra, pada tahun 1956 menemukan algoritma yang digunakan untuk menemukan jarak terpendek di antara *node-node* dalam suatu graf dengan nama algoritma Dijkstra.

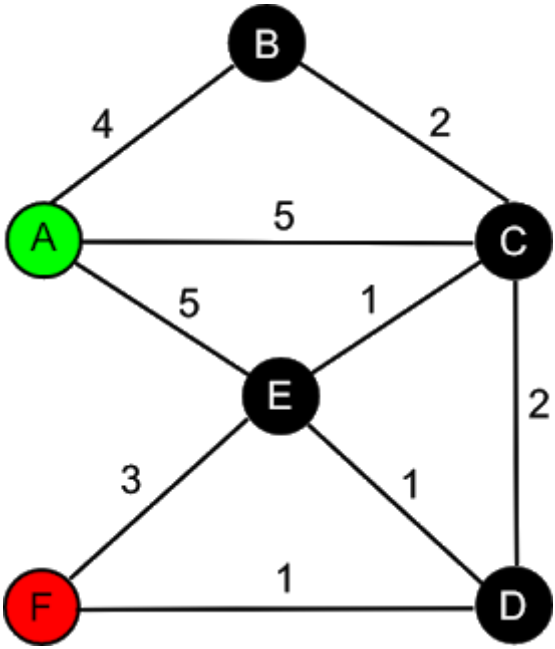
Algoritma Dijkstra ada dalam berbagai macam varian. Varian aslinya adalah algoritma pencari jalur terpendek antara dua *node* 3, tapi untuk varian yang ada pada umumnya sebuah *node* ditentukan sebagai *node* sumber dan dari sumber tersebut dicarilah jarak terpendek menuju *node-node* lainnya dalam satu grafik yang menghasilkan sebuah tree jarak terpendek.

Dari *node* sumber yang ada, algoritma menemukan jalur terpendek di antara tiap-tiap *node* yang ada 4. Ini dapat digunakan juga untuk mencari jarak terpendek dari sebuah *node* ke sebuah *node* tujuan dengan menghentikan algoritma ketika jarak terpendek menuju titik tujuan sudah didapat.

Untuk bisa menerapkan algoritma ini dibutuhkan beberapa data yang harus disiapkan, yaitu :

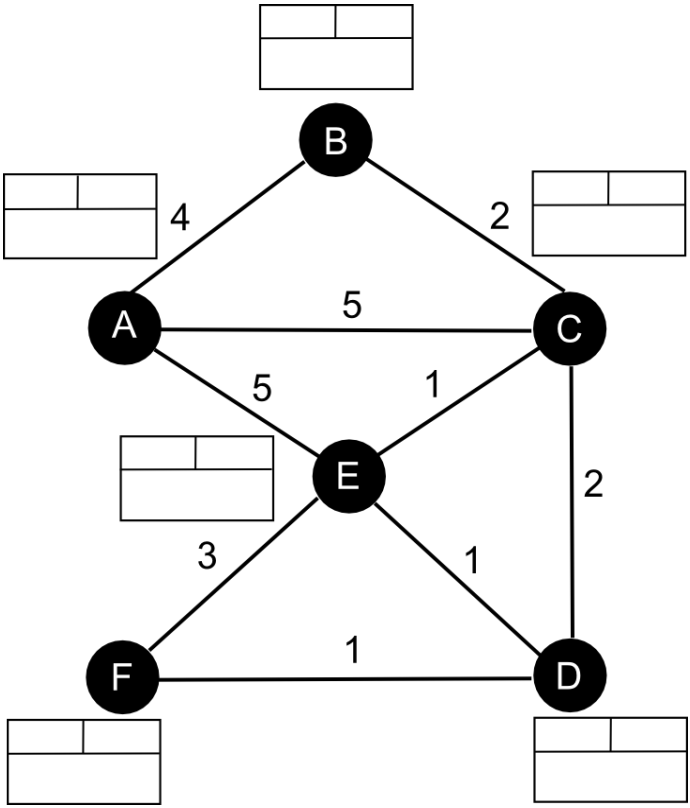
1. Beberapa Titik/simpul/daerah, titik/simpul/daerah yang bisa dijangkau secara langsung, dan juga jarak antara mereka.
2. Titik/simpul/daerah awal.
3. Titik/simpul/daerah tujuan.

Jika d*icon*tohkan dengan gambar grafik akan seperti ini :



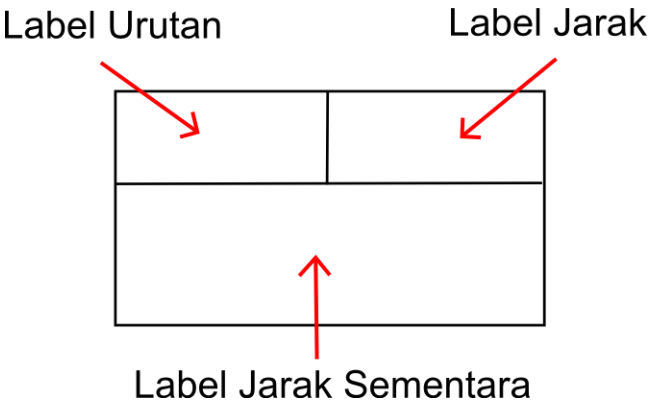
Gambar 2.1. Grafik peta titik simpul

Titik A adalah titik awal dan titik F adalah titik tujuan. Kemudian kita akan mencari rute manakah yang harus dilewati dan memilik total jarak yang paling dekat. Untuk bisa mendapatkan rute itu, maka grafik di atas ditambahkan beberapa kotak untuk mengisi beberapa label. Seperti ini :



Gambar 2.2. Grafik pelabelan peta titik

Penjelasannya adalah :



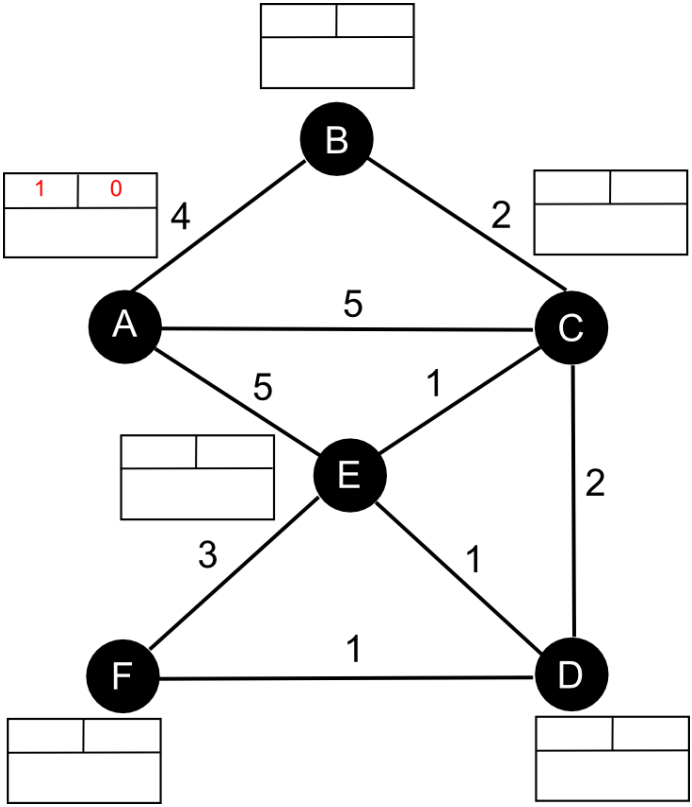
Gambar 2.3. Keterangan label grafik

Setelah itu ada beberapa langkah yang harus dilakukan, yaitu :

1. Mengisi kotak label pada titik awal dengan label urutan 1 dan label jarak 0.
2. Menetapkan label jarak sementara untuk semua titik yang dapat dihubungi langsung dari awal.
3. Pilih titik dengan label jarak sementara terkecil dan menuliskan nilainya di label jarak, serta tambahkan label urutannya.
4. Masukan label jarak sementara pada setiap titik yang belum memiliki label urutan dan label jarak dan dapat dihubungi langsung dari titik yang baru saja ditulis label jarak dan label urutannya. Nilainya diisi dengan total dari label jarak dari titik sebelumnya dan jarak dari titik tersebut. Jika label jarak sementara di titik tersebut sudah memiliki nilai, maka harus diganti hanya jika nilai yang baru lebih kecil.

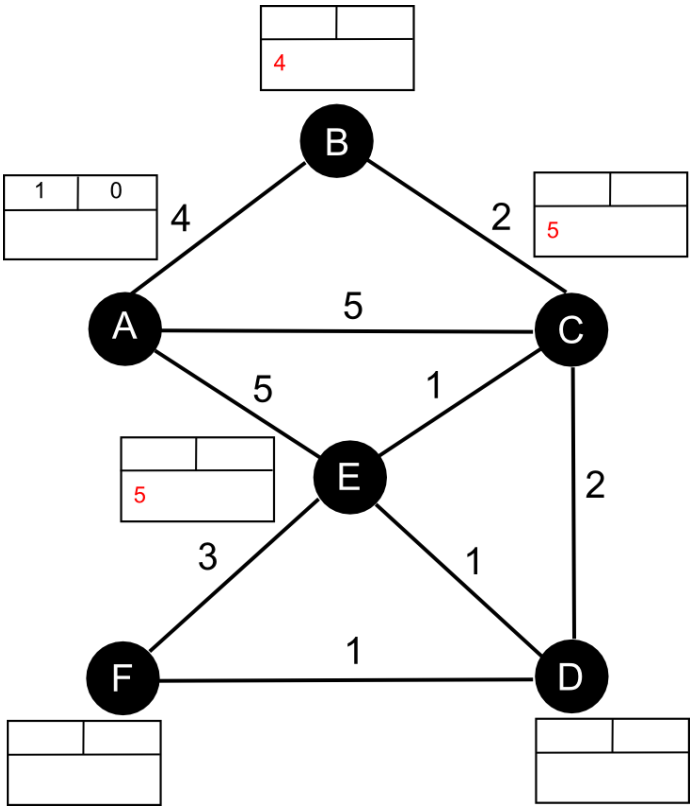
5. Pilih titik dengan label jarak sementara terkecil dan menggunakan label jarak sementaranya sebagai label jarak dari titik tersebut, serta tambahkan label urutannya.
6. Ulangi langkah 4 dan 5 hingga titik tujuan memiliki label jarak dan label urutan.

Maka pada langkah pertama adalah Mengisi kotak label pada titik awal dengan label urutan 1 dan label jarak 0.



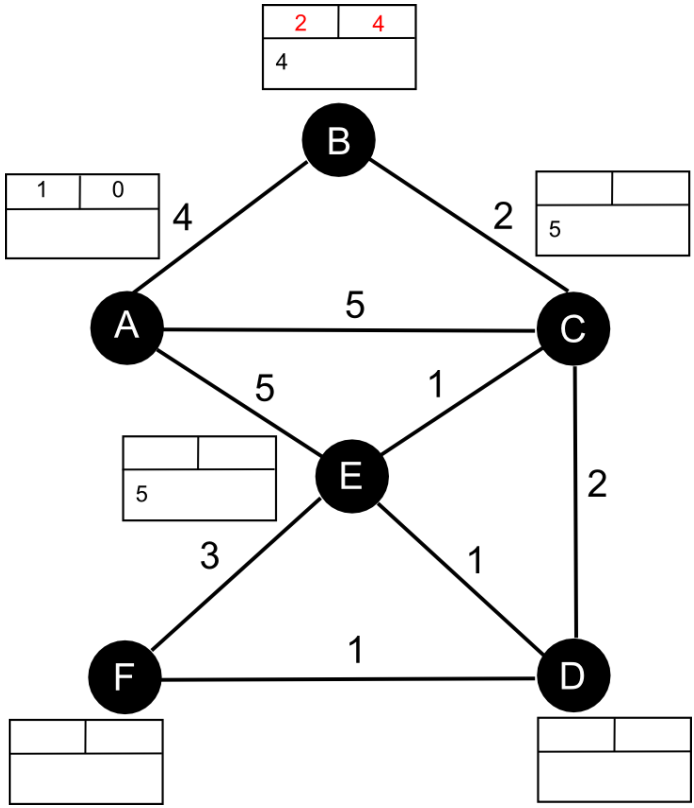
Gambar 2.4. Pengisian label iterasi pertama

Kemudian mengisi label jarak sementara titik yang dapat dihubungi langsung dari titik A yakni titik B, C, dan E.



Gambar 2.5. Pengisian label iterasi kedua

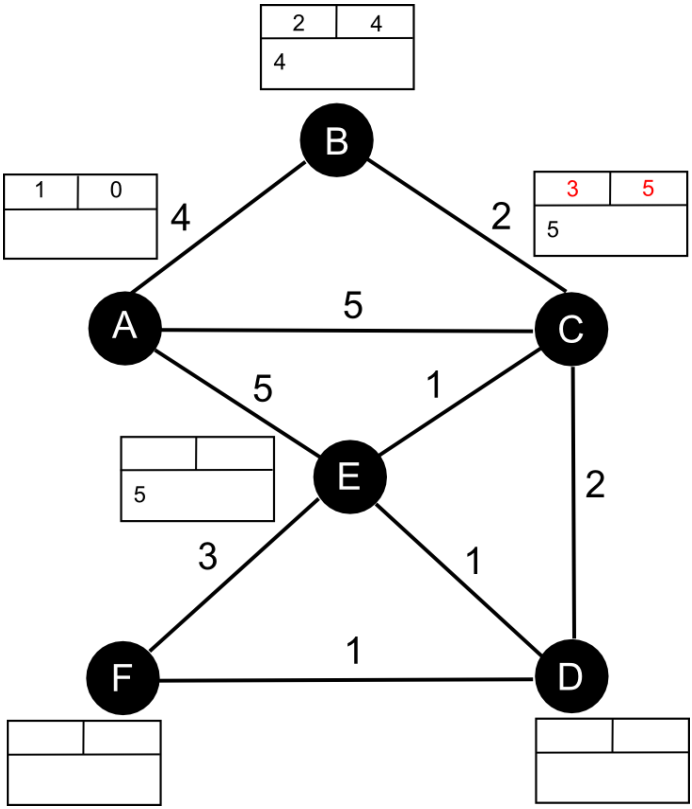
Maka yang terpilih adalah titik B karena memiliki label jarak sementara terkecil, dan mengisi nilai label jaraknya sama dengan label jarak sementara serta memberikan label urutannya.



Gambar 2.6. Pengisian label iterasi ketiga

Selanjutnya mengisi label jarak sementara titik yang belum memiliki label jarak dan dapat dihubungi langsung dari titik B yakni hanya titik C. Label jarak sementara titik C diisi dengan total jarak dari titik A sampai ke titik C yang melalui titik B, yakni 4 + 2 = 6. Namun sebelumnya nilai label jarak sementaranya titik C sudah ada dan lebih kecil (5), jadi label jarak sementaranya tidak diganti dan tetap bernilai 5.

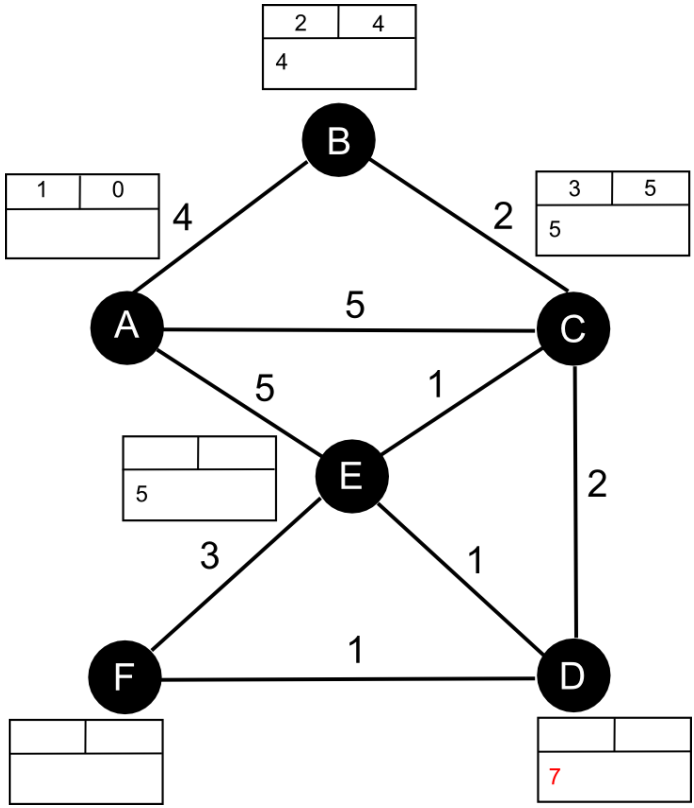
Langkah selanjutnya adalah memilih label jarak sementara terkecil. Karena titik E dan titik C memiliki label jarak sementara yang sama yakni 5, maka bisa memilih salah satu dari kedua titik tersebut. Misalkan titik C yang dipilih, maka berikan label jarak dan label urutannya.



Gambar 2.7. Pengisian label iterasi keempat

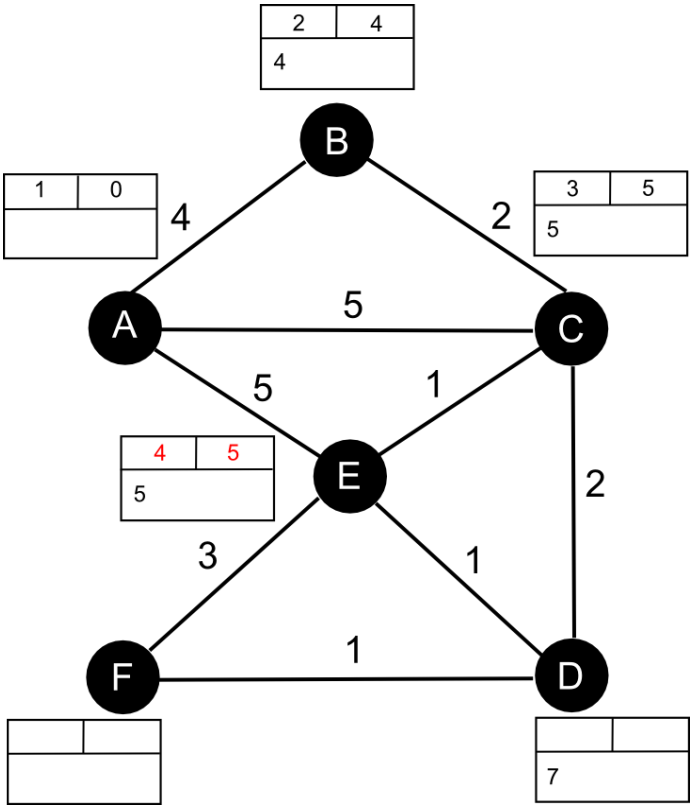
Kemudian titik yang dapat dihubungi secara langsung dari titik C dan belum memilik label jarak adalah titik E dan D. Titik E

=> 5 + 1 = 6, lebih besar jika dibandingkan dengan nilai label jarak sementara yang dimiliki oleh titik E sebelumnya (5), maka nilai 6 diabaikan dan tetap diisi 5. Titik D => 5 + 2 = 7, maka langsung saja label jarak sementara titik D diisi dengan 7.



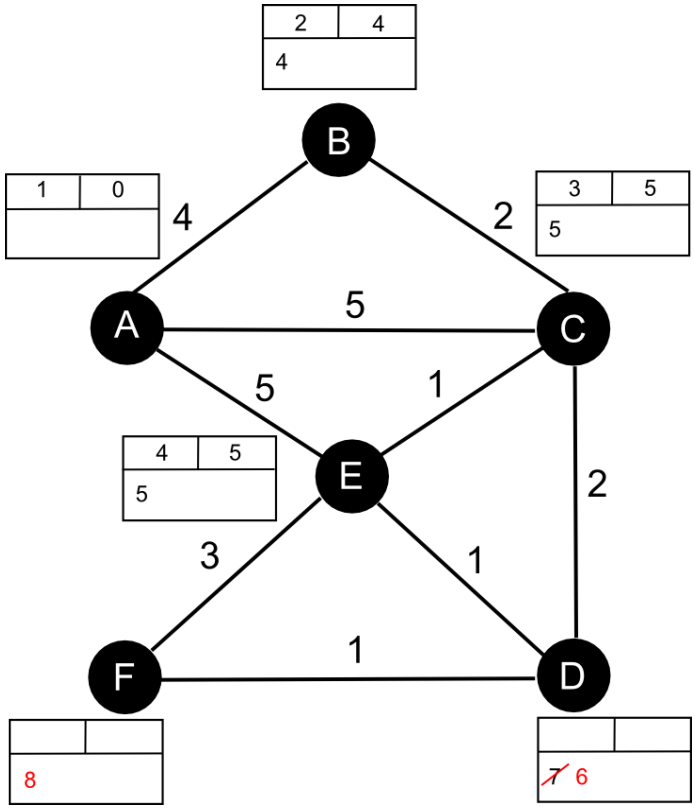
Gambar 2.8. Pengisian label iterasi kelima

Selanjutnya titik E terpilih karena memiliki label jarak sementara terkecil. Berikan label jarak dan label urutannya.



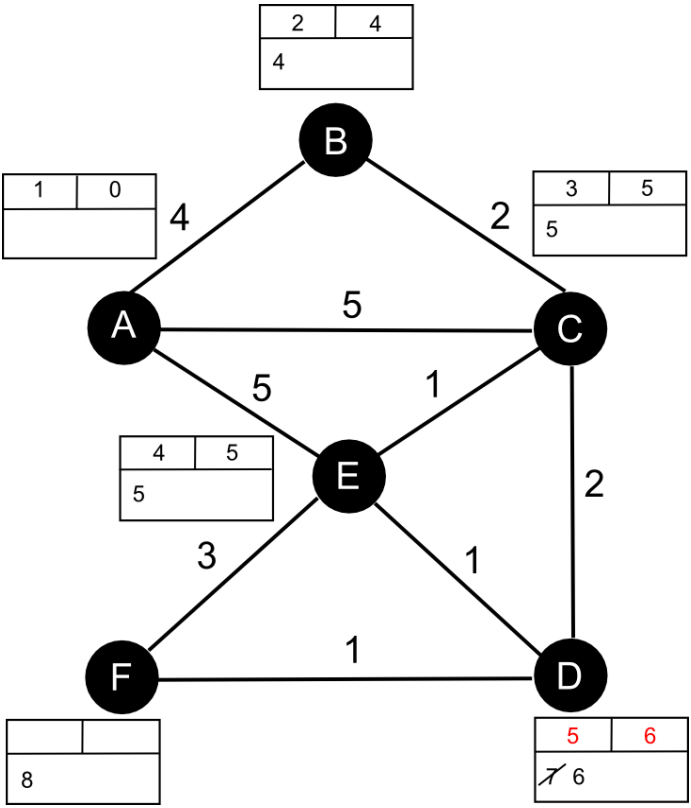
Gambar 2.9. Pengisian label iterasi keenam

Dan titik F dan G adalah titik yang dapat dihubungi secara langsung dari titik E dan belum memilik label jarak. Titik F => 5 + 3 = 8 dan langsung diisikan ke dalam label jarak sementaranya. sedangkan titik D => 5 + 1 = 6 dan lebih kecil dari pada nilai sebelumnya yaitu 7, maka nilai label jarak sementaranya diganti dengan 6.



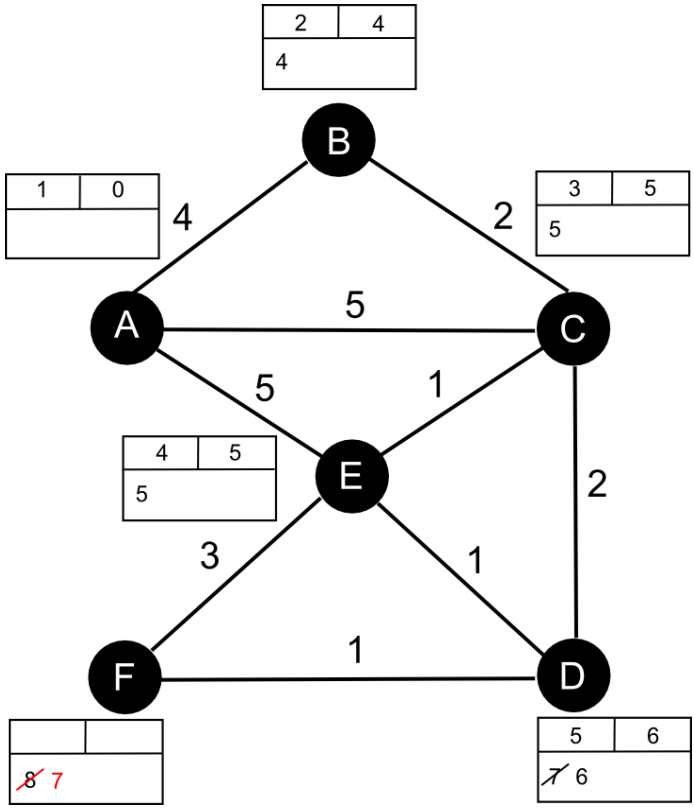
Gambar 2.10. Pengisian label iterasi ketujuh

Maka titik D terpilih karena memiliki label jarak sementara terkecil. Berikan label jarak dan label urutannya.



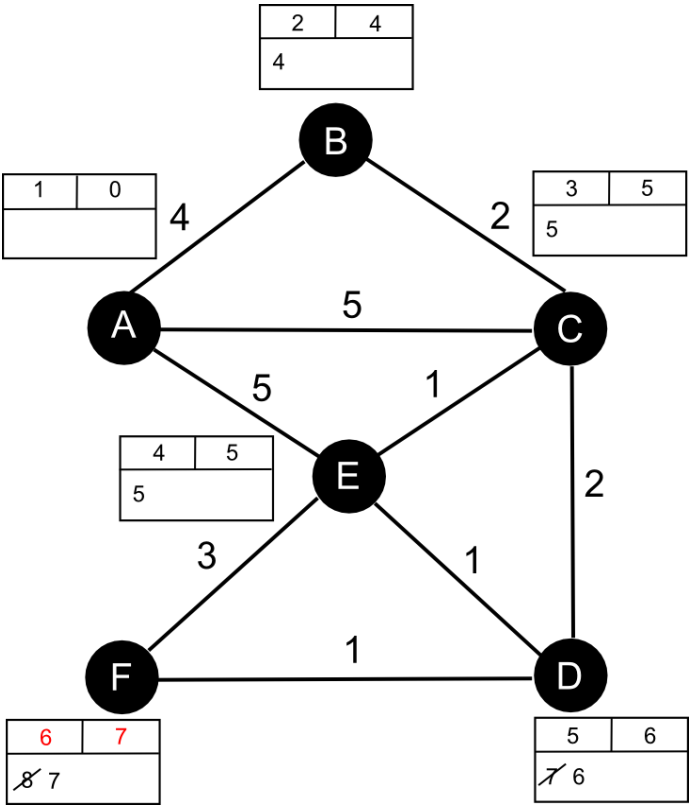
Gambar 2.11. Pengisian label iterasi kedelapan

Titik F adalah titik terakhir yang dapat dihubungi secara langsung dari titik D dan belum memilik label jarak serta merupakan titik tujuan. Titik F => 6 + 1 = 7 dan lebih kecil dari pada nilai sebelumnya yaitu 8, maka nilai label jarak sementaranya diganti dengan 7.



Gambar 2.12. Pengisian label iterasi kesembilan

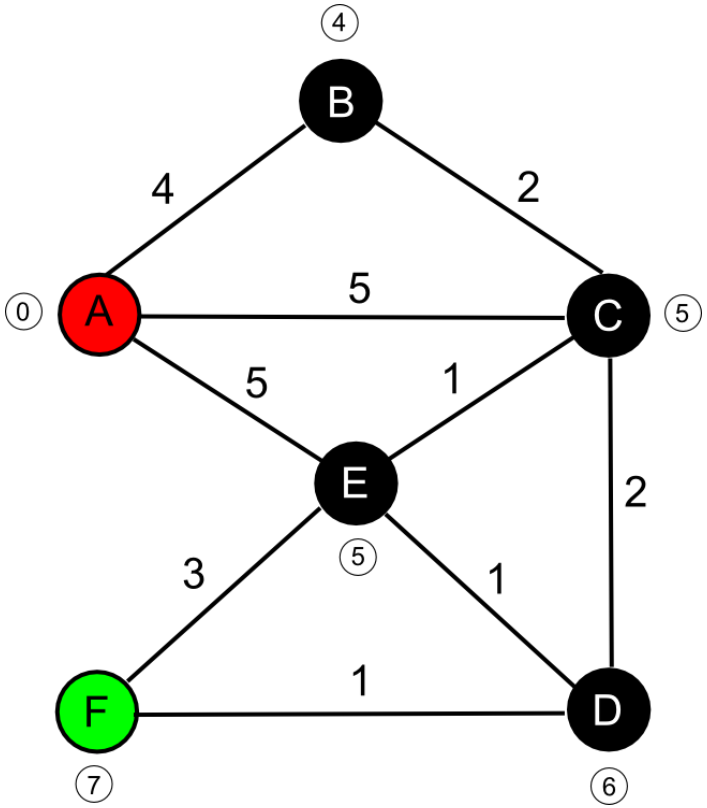
Karena titik F adalah satu-satunya titik terakhir yang belum mempunyai label jarak dan label urutan. maka langsung saja berikan nilai label jarak dan label urutannya. Dengan begitu titik tujuan sudah memiliki label jarak dan label jarak sementara.



Gambar 2.13. Pengisian label iterasi kesepuluh

Cara mengetahui rute yang harus dilewati

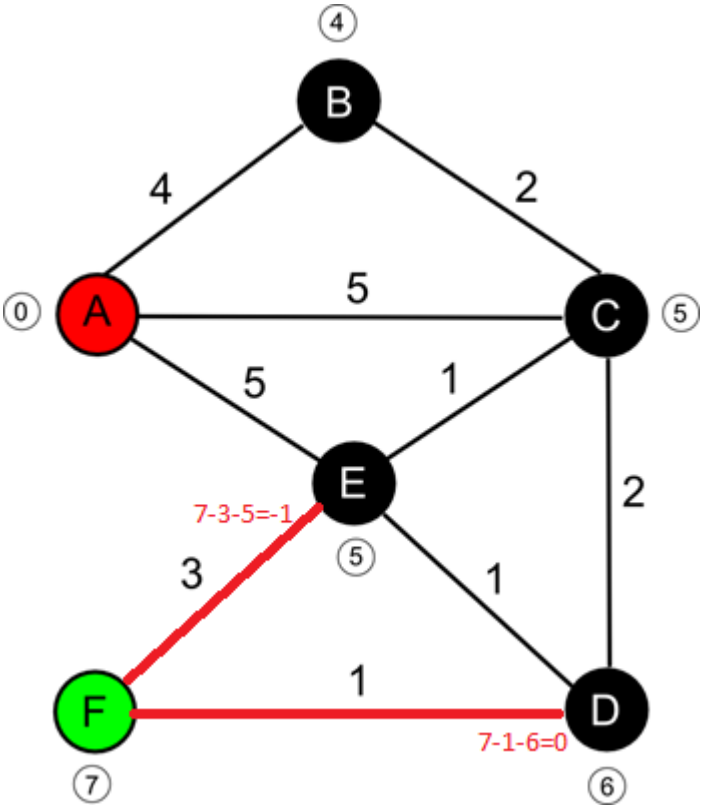
Untuk mengetahui rute manakah yang harus dilewati adalah dengan menelusuri kembali dari titik tujuan ke titik awal. tuliskan label jarak di samping setiap titik.



Gambar 2.14. Pemberian label jarak disetiap titik

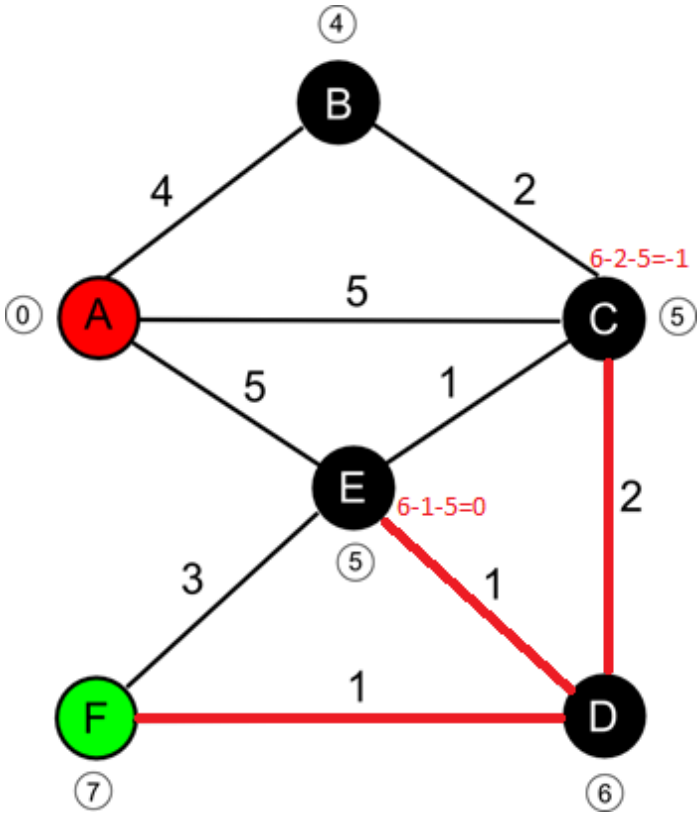
Titik mana sajakah yang dapat dihubungi langsung dari titik F ?, Yakni titik E dan D. maka, untuk menentukan titik manakah yang seharusnya dilewati adalah dengan cara mengurangkan label jarak titik F dengan jaraknya ke titik tujuan serta label jarak titik tersebut. jika hasilnya kurang dari 0 maka titik tersebut tidak layak untuk dilewati, dan jika hasilnya lebih dari 0 serta lebih mendekati 0 maka titik tersebut yang seharusnya dilewati.

Langkah pertama :



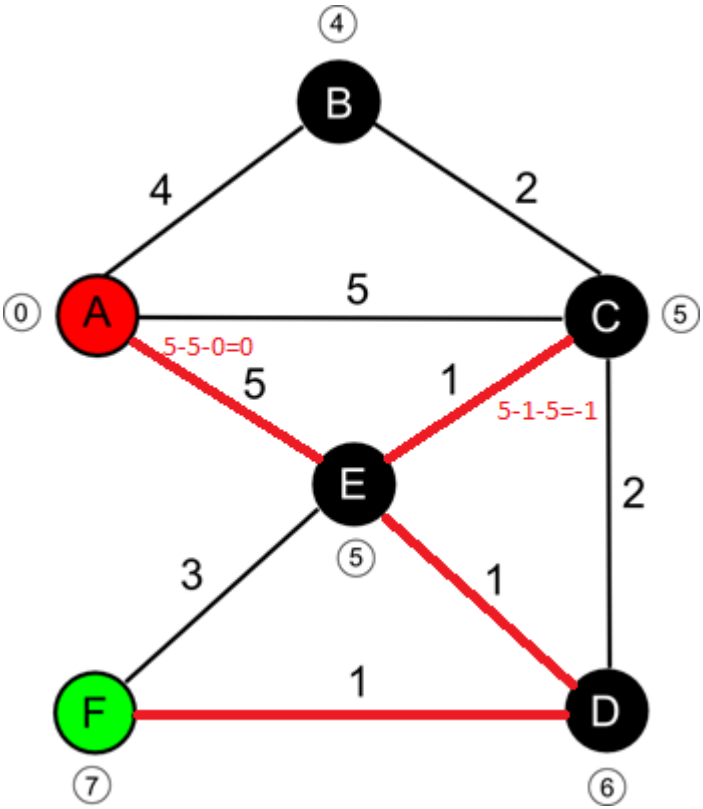
Gambar 2.15. Langkah pertama pencarian jalur

Langkah kedua :



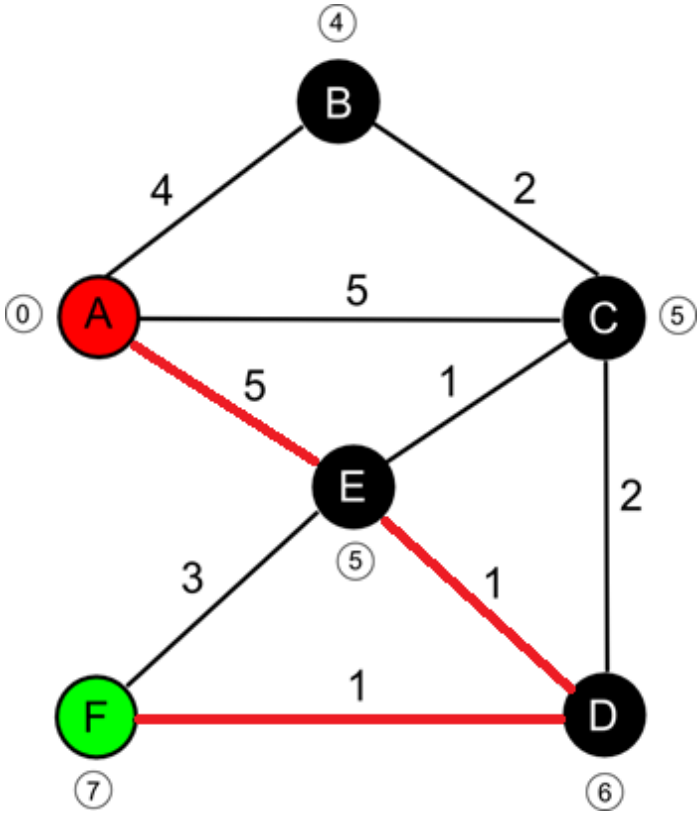
Gambar 2.16. Langkah kedua pencarian jalur

Langkah ketiga :



Gambar 2.17. Langkah ketiga pencarian jalur

Dengan begitu diketahui rute yang harus dilewati dan memiliki jarak terpendek dari titik A menuju titik F adalah A -> E -> D -> F



Gambar 2.18. Hasil akhir pencarian jalur dari itik awal ke titik akhir

2.2. GameMaker

Pada awalnya berjudul Animo, program ini pertama kali dirilis pada tahun 1999, dan mulai sebagai program untuk membuat animasi 2D. Nama itu kemudian diubah menjadi GameMaker, untuk menghindari konflik kekayaan intelektual dengan *software* 1991 Game-Maker. GameMaker diutamakan untuk membuat permainan yang menggunakan grafis 2D, yang memungkinkan

penggunaan grafis 3D yang terbatas. Mendukung kemampuan untuk membuat efek partikel seperti hujan, salju dan awan, namun tidak untuk *native* 3D kecuali melalui penggunaan *Dynamic Link Library*.

Untuk pemrograman yang lebih lanjut GameMaker dapat mengakomodasi dengan bahasa pemrograman *Game Maker Language*. *Game Maker Language* (GML) adalah bahasa *scripting* yang digunakan dalam GameMaker, yang biasanya secara signifikan lebih lambat dari bahasa dikompilasi seperti C ++ atau Delphi. Hal ini digunakan untuk lebih meningkatkan dan mengendalikan desain permainan melalui pemrograman yang lebih konvensional, yang bertentangan dengan *drag* dan drop sistem.

Komponen GameMaker

a. *Sprite*

Sprite merupakan visualisasi gambar yang akan digunakan untuk mempresentasikan objek pada game. *Sprite* dapat berupa gambar diam dan dapt juga gambar animasi.

Komponen *sprite*

- Kolom *Name* digunakan untuk mengisi nama *sprite* yang akan dibuat.
- Tombol Load *Sprite* digunakan untuk mengambil gambar yang akan dijadikan *sprite*.
- Tombol Edit *Sprite* digunakan untuk menjalankan editor membuat gambar.

b. Object

Merupakan objek yang digunakan sebagai fungsi variable game. Pada fungsi ini *sprite* yang telah dibuat akan didefinisikan dengan memberikan *event* dan *action*.

Komponen Object Properties

- Kolom *name* digunakan untuk mengisi nama objek
- Option *sprite* digunakan untuk memilih *sprite* yang akan dijadikan objek.
- *Add event* merupakan pilihan untuk memberikan *event* pada objek. Ada beberapa *event* yang dapat dipilih pada jendela *event selector*:
    - *Event create*
    - *Event create* berfungsi untuk memberikan perintah awal dari sebuah *action*
    - *Event destroy*
    - *Event destroy* berfungsi untuk mengakhiri atau menghentikan *event*.
    - *Event* alarm
    - *Event* alarm berfungsi untuk memberi jeda waktu
    - *Event* step
    - *Event* step berfungsi untuk memberikan perintah gerakan pada object.
    - *Event* collision
    - *Event* collision berfungsi untuk melakukan hubungan antara object yang satu dengan object yang lainya.
    - *Event Keyboard*
    - *Event keyboard* berfungsi untuk memberikan perintah kepada tombol-tombol yang ada pada *keyboard*.
    - *Event* Mouse

- *Event* mouse berfungsi untuk memberikan perintah pada mouse.
- *Event* Other
  Berisi banyak *event* seperti *view, game start, game end, room start, room end, no more live, no more health, animation end, end of path, close button, user definet, outside room* dan *intersect boundary*.
- *Event Draw*
  Event Draw berfungsi untuk menggambar health, live dan score.
- *Event Key Press*
  Event key press berfungsi untuk memberikan perintah kepada tombol-tombol yang ada pada *keyboard*.
- *Event Key Release*
  Event release berfungsi untuk memberikan perintah kepada tombol-tombol yang ada pada *keyboard* yang sudah di *release*.

- *Action* digunakan untuk memberikan beberapa perintah pada *event*, pada *tool action* terdapat sejumlah *icon* yang dapat digunakan sebagai *action*:
  - *icon* move : Berfungsi untuk membuat gerakan
  - *icon jump* : Berfungsi untuk membuat lompatan

- *icon* path : Berfungsi untuk membuat gerakan sesuai dengan path yang sudah dibuat
- e). *icon* object : Berfungsi untuk menambahakn object
- f). *icon* sound : Berfungsi untuk menambahkan sound
- g). *icon* room : Berfungsi untuk memberi *action* untuk berpindah ke room sebelumnya, sesudahnya maupun kembali ke room awal.
- h). *icon* score : Berfungsi untuk membuat score
- i). *icon* health : Berfungsi untuk membuat darah
- j). *icon* live : Berfungsi untuk membuat nyawa
- o). *icon draw*ing : Berfungsi untuk menggambar score, nyawa dan darah.

c. Background

Digunakan untuk memberikan tampilan pada game dalam bentuk gambar.

d. Room

Digunakan untuk menerapkan objek pada ruang game, room dapat juga disebut dengan level game.

GameMaker mengakomodasi *redistribusi* pada berbagai platform. Program ini dibangun untuk platform Windows, Windows 8, Mac OS X, Ubuntu, HTML5, Android, iOS, Windows Phone 8, Tizen, Xbox One, dan Playstation.

2.3. *Puzzle*

Menurut Patmonodewo (Misbach, Muzamil, 2010) kata *puzzle* berasal dari bahasa Inggris yang berarti teka-teki atau bongkar pasang, media *puzzle* merupakan media sederhana yang dimainkan dengan bongkar pasang.

Berdasarkan pengertian tentang media *puzzle*, maka dapat disimpulkan bahwa media *puzzle* merupakan alat permainan edukatif yang dapat merangsang kemampuan matematika yang dimainkan dengan cara membongkar pasang kepingan *puzzle* berdasarkan pasangannya.

Macam-macam *Puzzle*

Muzamil, Misbach (2010) menyatakan beberapa bentuk *puzzle*, yaitu:

2.3.1. *Puzzle* konstruksi

Puzzle rakitan (*construction puzzle*) merupakan kumpulan potongan-potongan yang terpisah, yang dapat digabungkan kembali menjadi beberapa model. Mainan rakitan yang paling umum adalah blok-blok kayu sederhana berwarna-warni. Mainan rakitan ini sesuai untuk anak yang suka bekerja dengan tangan, suka memecahkan *puzzle*, dan suka berimajinasi. Gambar 2.19 merupakan salah satu contoh *puzzle* kontruksi.

Gambar 2.19 *Puzzle* Konstruksi (sumber : http://permainananakmuslim.blogspot.com/2013/09/pengertian-macam-macam-dan-fungsi.html)

2.3.2. *Puzzle* batang (*stick*)

Puzzle batang merupakan permainan teka-teki matematika sederhana namun memerlukan pemikiran kritis dan penalaran yang baik untuk menyelesaikannya. *Puzzle* batang ada yang dimainkan dengan cara membuat bentuk sesuai yang kita inginkan ataupun menyusun gambar yang terdapat pada batang *puzzle*. Gambar 2.20 dan merupakan contoh *puzzle* batang *(stick)*

Gambar 2.20 *Puzzle* Batang (sumber : http://permainananakmuslim.blogspot.com/2013/09/pengertian-macam-macam-dan-fungsi.html)

2.3.3. *Puzzle* lantai

Puzzle lantai terbuat dari bahan *sponge* (karet/busa) sehingga baik untuk alas bermain anak dibandingkan harus bermain di atas keramik. *Puzzle* lantai memiliki desain yang sangat menarik dan tersedia banyak pilihan warna yang cemerlang. Juga dapat merangsang kreativitas dan melatih kemampuan berpikir anak. *Puzzle* lantai sangat mudah dibersihkan dan tahan lama. Gambar 2.21 merupakan salah satu contoh *puzzle* lantai.

Gambar 2.21 *Puzzle* Lantai (sumber : http://permainananakmuslim.blogspot.com/2013/09/pengertian-macam-macam-dan-fungsi.html)

2.3.4. *Puzzle* angka

Mainan ini bermanfaat untuk mengenalkan angka. Selain itu anak dapat melatih kemampuan berpikir logisnya dengan menyusun angka sesuai urutannya. Selain itu, *puzzle* angka bermanfaat untuk melatih koordinasi mata dengan tangan, melatih motorik halus serta menstimulasi kerja otak. Gambar 2.22 merupakan salah satu contoh *puzzle* angka

Gambar 2.22 *Puzzle* Angka (sumber : http://permainananakmuslim.blogspot.com/2013/09/pengertian-macam-macam-dan-fungsi.html)

2.3.5. *Puzzle* transportasi

Puzzle transportasi merupakan permainan bongkar pasang yang memiliki gambar berbagai macam kendaraan darat, laut dan udara. Fungsinya selain untuk melatih motorik anak, juga untuk stimulasi otak kanan dan otak kiri. Anak akan lebih mengetahui macam-macam kendaraan. Selain itu anak akan lebih kreatif, imajinatif dan cerdas. Gambar 2.23 merupakan salah satu contoh *puzzle* transportasi.

Gambar 2.23 *Puzzle* Transportasi (sumber : http://permainananakmuslim.blogspot.com/2013/09/pengertian-macam-macam-dan-fungsi.html)

2.3.6. *Puzzle* logika

Puzzle logika merupakan *puzzle* gambar yang dapat mengembangkan keterampilan serta anak akan berlatih untuk memecahkan masalah. *Puzzle* ini dimainkan dengan cara menyusun kepingan *puzzle* hingga membentuk suatu gambar yang utuh. Gambar 2.24 merupakan beberapa contoh *puzzle* logika.

Gambar 2.24 *Puzzle* Logika (sumber : http://permainananakmuslim.blogspot.com/2013/09/pengertian-macam-macam-dan-fungsi.html)

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN

Pada bab ini akan dibahas mengenai analisis sistem, perancangan sistem, perancangan perangkat lunak, dan implementasi perangkat lunak yang dibuat.

3.1. Analisis

Tahap analisis ini terbagi menjadi beberapa bagian antara lain: ranah permasalahan dan deskripsi umum perangkat lunak, arsitektur perangkat lunak dan spesifikasi kebutuhan perangkat lunak. Berikut penjabaran bagian-bagian tahap analisis.

3.1.1. Domain permasalahan

Permainan *puzzle* banyak dibuat menggunakan metode yang berbeda-beda. Permainan memiliki *puzzle* yang dibangkitkan secara otomatis secara acak oleh sistem dari beberapa kondisi yang diberikan. Permainan *puzzle* dapat menjadi sulit ataupun mudah bergantung dari bagaimana *puzzle* dibangkitkan oleh sistem. Namun terkadang *puzzle* yang dibangkitkan oleh sistem teramat sulit atau bahkan tidak dapat diselesaikan oleh pemain. Pembangkitan *puzzle* yang tidak dapat diselesaikan ini seharusnya memiliki mekanisme pengecek penyelesaian *puzzle* untuk memastikan bahwa semua level dapat diselesaikan.

Domain permasalahan dalam pembuatan permainan tattara akan dibatasi pada bagaimana penggunaan algoritma Dijkstra dalam menyelesaikan permainan *puzzle* tattara yang sudah dibangkitkan secara acak sehingga dapat dipastikan untuk

dapat diselesaikan pemain dan permainan tidak akan menemukan *puzzle* yang buntu saat dimainkan.

3.1.2. Deskripsi umum

Berdasarkan permasalahan yang ada pada pembahasan domain permasalahan, solusi yang ditawarkan adalah penggunaan algoritma Dijkstra sebagai pembangkit *puzzle* yang sudah dipastikan dapat diselesaikan oleh pemain. Pada pembangkitan *puzzle* yang secara acak dilakukan oleh sistem akan dilakukan pengecekan menggunakan algoritma Dijkstra untuk memastikan penyelesaian *puzzle* yang telah dibangkitkan. Pemilihan hasil pembangkitan *puzzle* yang secara acak dilakukan oleh sistem dipastikan tidak memakan waktu lama sehingga pemain tidak harus menunggu lama untuk memainkan *puzzle* yang dibuat. Pemain akan dapat menyelesaikan semua *puzzle* yang telah dibuat secara acak karena *puzzle* yang telah dibuat dipastikan memiliki penyelesaian.

3.1.3. Arsitektur sistem

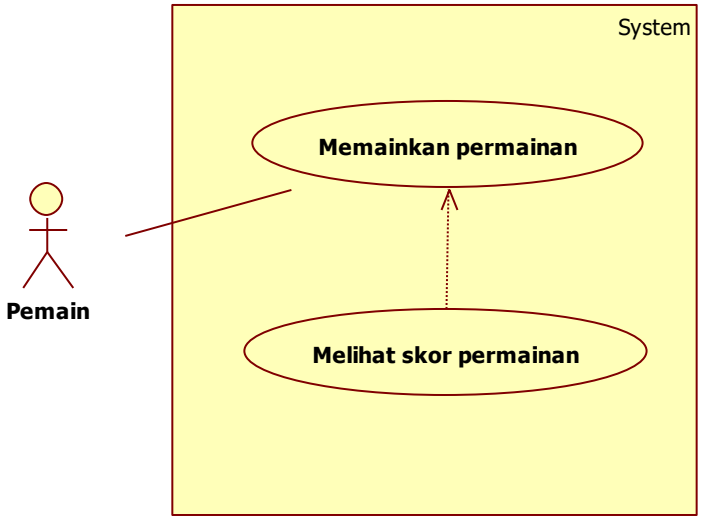
Dalam aplikasi yang dibuat pada tugas akhir ini, aplikasi permainan dapat dieksekusi menjadi aplikasi perangkat telepon pintar android maupun perangkat komputer atau laptop berbasis sistem operasi *windows*. Pada aplikasi permainan berbasis android maupun aplikasi permainan berbasis *windows* akan memiliki fungsi yang sama tanpa pembedaan pengurangan fungsi aplikasi permainan.

3.1.4. Aktor

Pada sistem yang dibuat ini, aktor yang dapat menjadi pengguna adalah semua orang. Pengguna aplikasi adalah orang yang dapat mengoperasikan telepon pintar berbasis android ataupun pengguna yang dapat mengoperasikan komputer atau laptop dengan sistem operasi *windows*. Pengguna dapat memilih untuk menjalankan aplikasi di perangkat yang dipilih sesuai keinginan pengguna.

3.1.5. Kasus pengguna

Pada subbab ini akan dijelaskan kasus penggunaan yang dibutuhkan pada sistem sesuai dengan analisa yang telah dilakukan. Diagram kasus penggunaan dapat dilihat pada Gambar 3.1 dan kode kasus penggunaan ada pada Tabel 3.1.



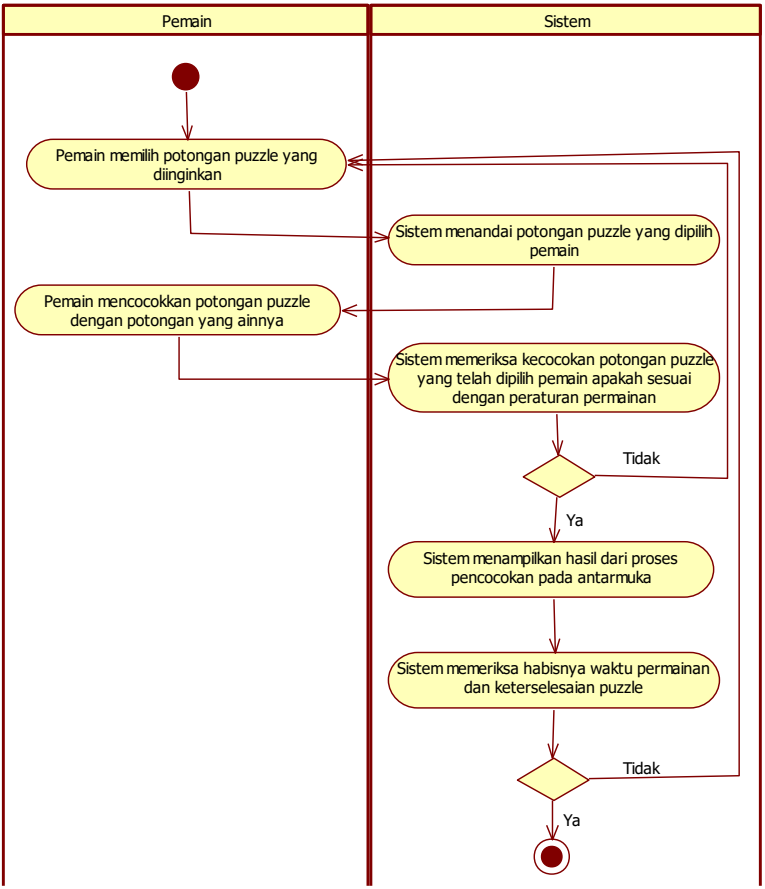
Gambar 3.1 *Use-case* Diagram

Tabel 3.1 keterangan Kode Kasus Penggunaan

Kode Kasus Penggunaan	Kasus Penggunaan
UC-001	Memainkan permainan
UC-002	Melihat skor permainan

3.1.5.1. Memainkan permainan

Pada kasus penggunaan ini sistem akan menerima masukan dari pemain dan akan memberikan umpan balik sebagai hasil dari masukan yang telah diberikan pemain. Pemain memberikan masukan pada permainan untuk menyelesaikan *puzzle* yang telah diberikan dan sistem akan mengolah masukan pemain telah sesuai dengan peraturan permainan yang telah ditentukan atau tidak.



Gambar 3.2 Diagram Aktivitas Memainkan Permainan

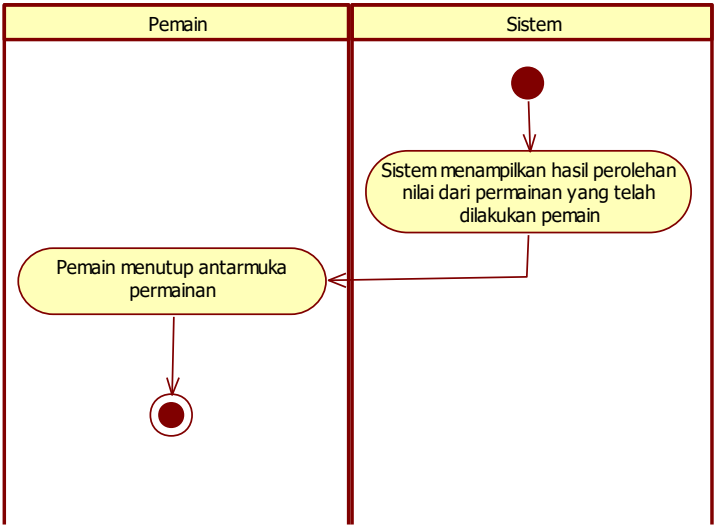
Tabel 3.2 Spesifikasi Kasus Penggunaan Memainkan Permainan

Nama	Memainkan permainan
Kode	UC-001
Deskripsi	Sistem menerima masukan dari pemain dan menampilkan hasil dari masukan pemain sesuai dengan peraturan permainan
Tipe	Fungsional
Pemicu	Pemain memilih potongan *puzzle* pada permainan
Aktor	Pemain
Kondisi Awal	Pemain berada pada antarmuka permainan
Aliran: - Kejadian Normal	1. Pemain memilih potongan *puzzle* yang diinginkan 2. Sistem menandai potongan *puzzle* yang dipilih pemain 3. Pemain mencocokkan potongan *puzzle* a dengan potongan yang lainnya 4. Sistem memeriksa kecocokan potongan *puzzle* yang telah dipilih pemain apakah sesuai dengan peraturan permainan

	5. Sistem menampilkan hasil dari proses pencocokan pada antarmuka 6. Sistem akan menghentikan permainan jika pemain telah selesai menyelesaikan seluruh *puzzle* atau waktu telah habis
- Kejadian Alternatif	-
Kondisi Akhir	Permainan berakhir

3.1.5.2. Melihat skor permainan

Pada kasus penggunaan ini sistem akan menampilkan hasil perolehan nilai dari permainan yang telah dilakukan oleh pemain.



Gambar 3.3 Melihat Skor Permainan

Tabel 3.3 Spesifikasi Kasus Penggunaan Melihat Skor Permainan

Nama	Melihat Skor Permainan
Kode	UC-001
Deskripsi	Sistem menampilkan hasil perolehan nilai permainan pemain
Tipe	Fungsional
Pemicu	Pemain telah menyelesaikan permainan atau waktu permainan telah habis
Aktor	-
Kondisi Awal	Permainan telah berakhir

Aliran: - Kejadian Normal	1. Sistem menampilkan hasil perolehan nilai dari permainan yang telah dilakukan pemain 2. Pemain menutup antarmuka permainan
- Kejadian Alternatif	-
Kondisi Akhir	Pemain keluar dari permainan

3.1.6. Spesifikasi Kebutuhan Perangkat Lunak

Bagian ini berisi tentang kebutuhan perangkat lunak. Kebutuhan perangkat lunak dalam sistem ini mencakup kebutuhan fungsional saja. Pada bab ini juga dijelaskan tentang spesifikasi terperinci pada masing-masing kebutuhan fungsional. Rincian spesifikasi dari kasus penggunaan disajikan dalam bentuk tabel.

3.1.6.1. Kebutuhan fungsional sistem

Kebutuhan fungsional berisikan proses-proses yang dibutuhkan dalam sistem dan harus dijalankan. Kebutuhan fungsional sistem dideskripsikan dalam berikut.

Tabel 3.4 Kebutuhan Fungsionalitas Sistem

Kode Kebutuhan	Kebutuhan Fungsionalitas	Deskripsi
F-001	*Multiplatform*	Aplikasi permainan dapat berjalan pada platform mobile maupun desktop.

3.1.6.2. Kebutuhan non fungsional sistem

Kebutuhan non fungsional berisikan batasan-batasan ataupun fitur pada sistem diluar kebutuhan fungsional. Kebutuhan non fungsional sistem dideskripsikan dalam tabel berikut.

Tabel 3.5 Kebutuhan Non Fungsional

Kode Kebutuhan	Kebutuhan Non Fungsional	Deskripsi
NF-001	*Capability*	Aplikasi permainan memiliki antarmuka yang sederhana dan mudah dimainkan semua orang
NF-002	*Compability*	Aplikasi memiliki ukuran penggunaan penyimpanan yang kecil

3.2. Perancangan Sistem

Penjelasan tahap perancangan perangkat lunak dibagi menjadi beberapa bagian yaitu perancangan diagram kelas, perancangan proses analisis, dan perancangan antarmuka.

3.2.1. Perancangan algoritma

Metode yang digunakan dalam pembuatan permainan ini menggunakan kode berbasis skrip. Pada pembuatan permainan algoritma yang akan digunakan adalah algoritma Dijkstra yang dikemukakan oleh ilmuwan yang bernama Edsger Dijkstra. Algoritma tersebut dapat menentukan bobot dari masing-masing potongan *puzzle* yang bersebelahan yang nantinya akan digunakan untuk pencocokan. Tahapan penggunaan algoritma adalah sebagai berikut.

Pertama-tama tentukan titik mana yang akan menjadikan *node* awal, lalu beri bobot jarak pada *node* pertama ke *node* terdekat satu persatu, Dijkstra akan melakukan pengembangan pencarian dari satu titik ke titik lain dan ke titik selanjutnya tahap demi tahap inilah urutan logika dari algoritma Dijkstra :

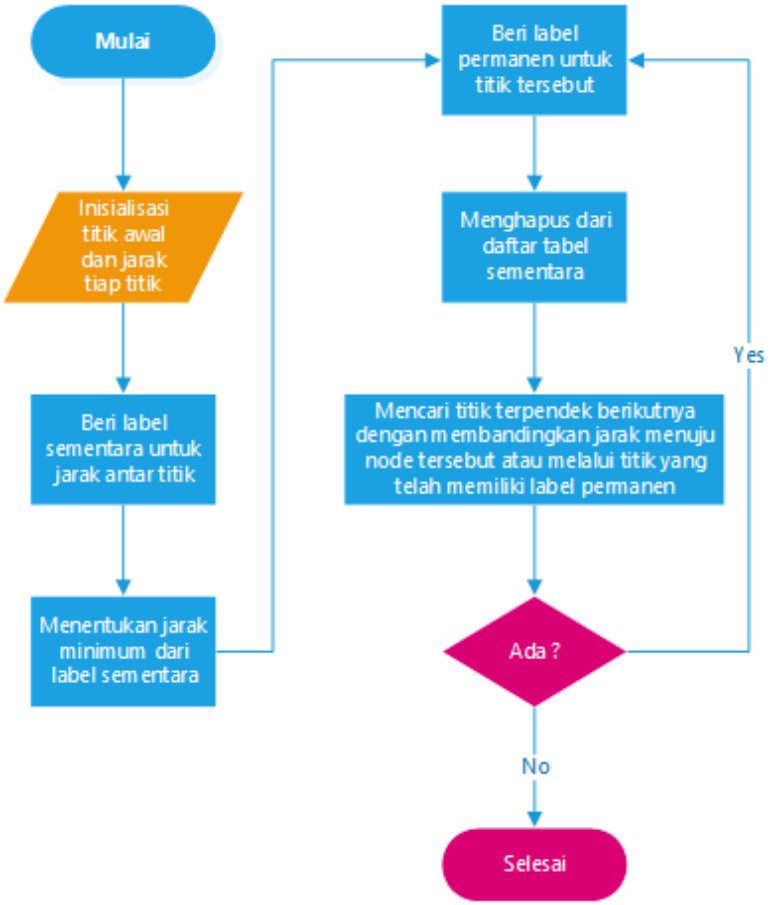
1. Beri nilai bobot (jarak) untuk setiap titik *puzzle* ke titik *puzzle* lainnya, lalu set nilai 0 pada *node puzzle* awal dan nilai tak hingga terhadap *node puzzle* lain (belum terisi)
2. Set semua *node puzzle* dengan jarak tak terbatas dan set *node puzzle* awal dengan nilai minimal.
3. Dari *node puzzle* awal, pertimbangkan *node puzzle* tetangga yang belum terhitung dan hitung jaraknya dari titik awal. Jika jarak node *puzzle* dengan tetangga yang baru lebih kecil dari jarak *node puzzle* dengan *node puzzle* tetangga yang

telah terekam sebelumnya maka hapus data lama, simpan ulang data jarak dengan jarak yang baru.

4. Saat selesai mempertimbangkan setiap jarak terhadap *node puzzle* tetangga, tandai *node puzzle* yang telah dikunjungi. *Node puzzle* yang telah dikunjung tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
5. Atur *node puzzle* yang belum dikunjungi dengan jarak terkecil dari *node* awal sebagai *node awal* yang baru selanjutnya dan lanjutkan dengan kembali ke step 3

Pada setiap node puzzle akan diberikan bobot-bobot yang akan digunakan untuk menentukan besarnya skor. Skor yang akan diperoleh jika pengguna berhasil menyelesaikan potongan puzzle. Penambahan besar skor pengguna didasarnya oleh banyaknya runtutan puzzle yang dapat diselesaikan, semakin banyak runtutan puzzle yang diselesaikan pengguna maka semakin besar pula skor yang akan diperoleh pengguna.

Pemberian skor yang akan didapat pengguna akan dinilai berdasar dengan kecepatan pengguna, semakin cepat pengguna menyelesaikan puzzle semakin banyak pula skor yang akan didapatkan oleh pengguna. Skor yang didapat bersifat linear sehingga pengguna tidak mendapatkan kombo jika menyelesaikan puzzle lain setelah menyelesaikan satu potongan puzzle secara beruntun ataupun dalam satu waktu tertentu.



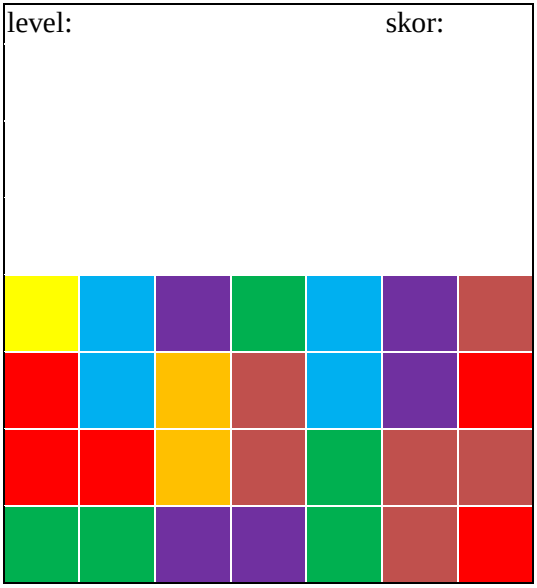
Gambar 3.4. *Flowchart* algoritma dijkstra

3.2.2. Perancangan antarmuka

Pada subbab ini akan dijelaskan perancangan antarmuka sistem yang akan dibuat.

Antarmuka akan dibuat dalam bentuk aplikasi mobile dan juga aplikasi desktop. Antarmuka aplikasi

desktop dan mobile tidak akan dibedakan hanya resolusi antarmuka pada perangkat mobile akan disesuaikan dengan perangkat mobile di mana aplikasi akan dijalankan. Untuk pengaturan aplikasi yang lain akan diatur saat aplikasi akan diluncurkan pada perangkat mobile untuk memudahkan penyesuaian dengan perangkat mobile yang digunakan. Berikut contoh antarmuka permainan tattara.



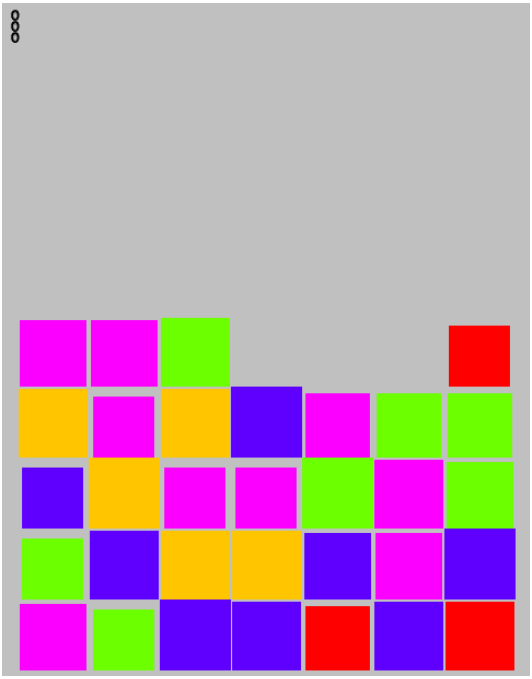
Gambar 3.5 Rancangan Antarmuka Aplikasi

BAB IV
IMPLEMENTASI

Pada bab ini dijelaskan implementasi sesuai dengan desain yang telah ditentukan sebelumnya.

4.1. Implementasi Antarmuka

Pada subbab ini akan dijelaskan tentang implementasi antarmuka yang menjadi bagian terluar sekaligus bagian yang akan berinteraksi langsung dengan pengguna. Pada antarmuka permainan pengguna dihadapkan langsung dengan permainan dan menyelesaikannya sampai permainan berakhir.



Gambar 4.1 Antarmuka Permainan Tattara

4.2. Implementasi Fitur

Sistem permainan tattara saat dijalankan memproduksi potongan-potongan *puzzle* secara acak dan menyusunnya dalam tumpukan untuk diselesaikan pengguna. Sistem akan mengacak dari 6 buah potongan *puzzle* berbeda warna menjadi satu baris setiap 6 detik. Pemindahan potongan *puzzle* juga memiliki batas waktu yakni hanya 3 detik. Jika potongan tidak diletakkan pada tempatnya maka potongan yang sedang dipindahkan akan kembali ke tempatnya semula.

Setiap *puzzle* yang dipilih untuk diselesaikan akan mengubah posisi *puzzle* lain disekitarnya. Oleh karena itu pemain harus memperhatikan posisi potongan-potongan *puzzle* lain disekitar untuk menyelesaikan *puzzle* sehingga akan mendapat skor yang tinggi. Untuk mendapatkan skor yang tinggi pemain harus membuat potongan *puzzle* yang semakin panjang dan menyelesaikannya dalam satu kali pemindahan potongan *puzzle*.

Pada algoritma dijkstra terdapat bobot untuk menentukan besaran nilai yang akan dikeluarkan sebagai skor dalam permainan. Pemberian bobot bersifat konstan sehingga nilai yang didapatkan pengguna juga akan bertambah secara konstan tidak terpengaruh oleh kombo yang dihasilkan. Nilai pembobotan dideklarasikan pada variabel global dan akan dipanggil untuk memberikan skor pada fungsi *alarm 1* dalam objek *spriteObject*. Kemudian skor akan diperbarui setiap pemain menyelesaikan potongan *puzzle* sejenis. Pada permainan skor ditampilkan dari fungsi *draw* dalam objek *controlObject* untuk ditampilkan ke dalam antarmuka permainan.

```
var level = argument0;
var gems_instance = argument1;

switch(level)
{
    case 1:
        var i=irandom(5);
        gems_instance.gemsType = i;
        gems_instance.image_speed    =
random(0.1)+0.3;
        gems_instance.uniqId         =
gems_instance;
        if(i==0){

gems_instance.sprite_index = sp_gem1;
            return 0;
        }else if(i==1){

gems_instance.sprite_index = sp_gem2;
            return 0;
        }else if(i==2){

gems_instance.sprite_index = sp_gem3;
            return 0;
        }else if(i==3){

gems_instance.sprite_index = sp_gem4;
            return 0;
        }else if(i==4){

gems_instance.sprite_index = sp_gem5;
            return 0;
        }else if(i==5){
```

```
gems_instance.sprite_index = sp_gem6;
            return 0;
        }
    break;
}
```

Kode Sumber 4.1 Baris kode pengacakan potongan *puzzle*

```
if(            uniqId!=-1            &&
global.pickupId==uniqId ){

draw_sprite(sp_gotoback,0,global.goto
backX,global.gotobackY);
}
draw_self();

if(            uniqId!=-1            &&
global.pickupId==uniqId ){

draw_sprite_part(sp_release,0,0,0,glo
bal.spReleaseWidth,

global.spReleaseHeight,x,y);

draw_sprite_part(sp_release,1,0,0,((g
lobal.holdTime-(global.holdTime-
alarm[2]))*global.spReleaseWidth/glob
al.holdTime),

global.spReleaseHeight,x,y);
}
```

Kode Sumber 4.2 Baris kode penggambaran objek potongan *puzzle* pada antarmuka permainan

```
mousePressed = false;
gemsType = -1;
oldX = -1;
oldY = -1;
uniqId = -1;
havetoMoveX = -1;
havetoMoveY = -1;
havetoMove = false;
destory = false;
```

Kode Sumber 4.3 Baris kode inisiasi objek potongan *puzzle*

```
if(global.currentLevel==1){
    countH = 4;
    global.genBlockTime = 360;
    alarm[0] = global.genBlockTime;
}

for(i=0;i<countH;i++){

for(j=0;j<global.gameBlockWidth;j++){
        var            ins            =
instance_create(global.gameStartX+j*g
lobal.blockSize,global.gameStartY-
i*global.blockSize,obj_gems);

get_random_sprite_index(global.curren
tLevel,ins);
    }
}
```

Kode Sumber 4.4 Baris kode inisiasi objek kontrol permainan

```
draw_text(10,10,global.dropping);
draw_text(10,20,global.holding);
draw_text(10,30,global.moving);
draw_set_color(c_white);
draw_text(220, 20,
string_lettersdigits("Score: " +
string(global.point)));
```

Kode Sumber 4.5 Baris kode penggambaran kontrol permainan

```
instance_destroy();

global.point += global.pointGet;
```

Kode Sumber 4.6 Baris kode pemberian bobot pada skor permainan

BAB V
PENGUJIAN DAN EVALUASI

Pada bab ini dijelaskan tentang uji coba dan evaluasi dari implementasi yang dilakukan pada tugas akhir ini.

5.1. Lingkungan Uji Coba

Pengujian tugas akhir ini dilakukan pada perangkat yang ada pada tabel berikut

Tabel 5.1 Lingkungan Pengujian Perangkat Lunak

Jenis Perangkat	Perangkat Selular	Laptop
Prosesor	Quad-core, 1300MHz	Intel Core i3217U CPU @ 1800MHz
Memori	1 GB	3 GB
Sistem Operasi	Android	*Windows*
Jenis Sistem Operasi	5.0 (Lolipop)	*Windows* 8.1 (x64)

5.2. Skenario Uji Coba

Pada subbab ini akan dijelaskan uji coba yang dilakukan. Pengujian dilakukan terhadap aplikasi permainan dengan dilakukan uji fungsional. Pengujian dilakukan kepada pengguna dengan skenario pengguna memainkan permainan dalam satu kali permainan. Setelah melakukan pengujian aplikasi pengguna melakukan pengisian kuesioner, untuk mengetahui tanggapan pengguna terhadap sistem permainan

yang digunakan. Daftar responden dan pertanyaan yang diajukan kepada pengguna dapat dilihat pada tabel berikut.

Tabel 5.2 Data Penguji

No.	Nama	Usia
1	M. Ricky	22
2	Syamsuri	47
3	Rega Kusuma	29
4	Satrio Agung	25
5	Rega Kusuma	28
6	Jimmy Restu	22
7	Ahmad Putra Hidayah	26
8	Yunus Almakhlufi	20
9	Edwin Cahyadi	29
10	Eko Saputro	23
11	Andika Purnomo	23

Tabel 5.3 Daftar Pernyataan dan Hasil Kuesioner

No.	Pernyataan	Hasil Kuesioner
1	Aplikasi mempunyai tampilan yang menarik	63,63636364 %
2	Aplikasi mudah digunakan	79,54545455 %
3	Aplikasi mempunyai tingkat kesulitan yang mudah	72,72727273 %
4	Aplikasi mempunyai tanggapan yang cepat	52,27272727 %

Tabel 5.4 Keterangan Skala Kesesuaian

No.	Skala	Keterangan
1	4	Sangat setuju
2	3	Setuju
3	2	Tidak setuju
4	1	Sangat tidak setuju

Tabel 5.5. Hasil kuesioner pengguna

No.	1	2	3	4	5	6	7	8	9	10	11
Aplikasi mempunyai tampilan yang menarik	3	3	1	1	2	4	3	4	4	1	2
Aplikasi mudah digunakan	4	3	4	2	3	3	2	3	4	3	4
Aplikasi mempunyai tingkat kesulitan yang mudah	1	1	2	4	4	3	4	4	4	4	1
Aplikasi mempunyai tanggapan yang cepat	3	2	3	1	1	1	2	4	1	2	3

BAB VI
PENGUJIAN DAN EVALUASI

Pada bab ini dijelaskan mengenai kesimpulan dari hasil uji coba yang telah dilakukan dan saran mengenai hal-hal yang masih bisa untuk dikembangkan dari tugas akhir ini.

6.1. Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi dan pengujian aplikasi yang digunakan dapat diambil kesimpulan sebagai berikut

1. Aplikasi dapat digunakan dengan mudah oleh berbagai pengguna dengan usia yang beragam dengan dibuktikan dari survei yang menyatakan 79,5% setuju pada pernyataan tersebut.
2. Aplikasi memiliki tingkat kesulitan yang mudah diselesaikan oleh berbagai pengguna dengan hasil survei 72,7% pengguna setuju pada pernyataan.

6.2. Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang, berdasarkan pada hasil perancangan, implementasi dan uji coba yang telah dilakukan.

1. Aplikasi dapat digunakan sebagai sarana permainan yang dapat melatih kecepatan pengguna dalam mengambil keputusan dengan menyelesaikan permainan dengan cepat.
2. Aplikasi dapat dijalankan pada perangkat dengan spesifikasi yang rendah sekalipun tanpa kendala.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

Djikstra, E. W. (1959). *Numerische Mathematik.* Springer-Verlag[online].

Mehlhorn, K., & Sanders, P. (2008). *Algorithms and Data Structures: The Basic Toolbox.* Springer.

Rollings, A., & Adams, E. (2003). *Andrew Rollings and Ernest Adams on Game Design. New Rider*.

"Game Maker *Studio engine*". Desura. 29 May 2013[online].

McHugh, JA., *Algorithmic Graph Theory*, *Prentice Hall International*, *Inc*., *Englewood Cliffs*, NJ., 1990

Brassard, G., dan Bratley, P., *Fundamentals of Algorithmics*, *Prentice Hall International*, *Inc*., Singapore, 1996

B. Liu, S.-H. Choo, and S.-L. Lok, "*Finding the Shortest Route Using Cases, Knowledge, and Dijkstra's Algorithm*," CAIA, vol. 1994

K. R. Rao, "*Design & Analisys of Algorithms - In Simple Way,"* KL Univ., vol. 2010

[Halaman ini sengaja dikosongkan]

LAMPIRAN

```
var level = argument0;
var gems_instance = argument1;

switch(level)
{
    case 1:
        var i=irandom(5);
        gems_instance.gemsType = i;
        gems_instance.image_speed = random(0.1)+0.3;
        gems_instance.uniqId = gems_instance;
        if(i==0){
            gems_instance.sprite_index = spriteRed;
            return 0;
        }else if(i==1){
            gems_instance.sprite_index = spriteYellow;
            return 0;
        }else if(i==2){
            gems_instance.sprite_index = spriteGreen;
            return 0;
        }else if(i==3){
            gems_instance.sprite_index = spriteBlue;
            return 0;
        }else if(i==4){
            gems_instance.sprite_index = spritePurple;
            return 0;
        }else if(i==5){
            gems_instance.sprite_index = spritePink;
            return 0;
        }
    break;
}
```

Kode Sumber 1 ***Script randomizeSprite***

```
global.goBackX = -1;
global.goBackY = -1;
global.pickupId = -1;
global.uniqId = 0;

global.holding = false;
global.moving = false;
global.dropping = true;
global.needToMatchCheck = true;

global.blockSize = 64;
global.onceMoveStep = 16;
global.width = 480;
global.height = 640;
global.gameStartY = global.height-global.blockSize;
global.gameStartX = 50;
global.gameBlockWidth = 7;
global.gameBlockHeight = 9;
global.spriteReleaseWidth =
sprite_get_width(spriteRelease);
global.spriteReleaseHeight =
sprite_get_height(spriteRelease);
global.gameover = false;

global.currentLevel = 1;

global.holdTime = 60;

global.genBlockTime = 240;
global.genBlock = false;

global.drop = false;

global.point = 0;
global.pointGet = 7;

global.INFINITY = 9999;
global.MAX = 10;

room_goto_next();
```

Kode Sumber 2 *Sript startGame*

```
var x1 = argument0
var y1 = argument1
var x2 = argument2
var y2 = argument3

return ((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2))
```

Kode Sumber 3 ***Script getDistance***

```
var blocksInfo = argument0;
var i,j;
var is1,is2,is3;

for(j=0;j<global.gameBlockHeight;j++){
    for(i=0;i<global.gameBlockWidth-2;i++){
        is1 = blocksInfo[i+j*global.gameBlockWidth];
        is2 = blocksInfo[i+1+j*global.gameBlockWidth];
        is3 = blocksInfo[i+2+j*global.gameBlockWidth];
        if( is1!= noone && is2!= noone && is3!=
noone ){
            if(( is2.gemsType ==
is1.gemsType )&&( is3.gemsType == is1.gemsType )){
                is1.alarm[0] = 1;
                is1.destory = true;
                is2.alarm[0] = 1;
                is2.destory = true;
                is3.alarm[0] = 1;
                is3.destory = true;
            }
        }
    }
}

for(j=0;j<global.gameBlockHeight-2;j++){
    for(i=0;i<global.gameBlockWidth;i++){
        is1 = blocksInfo[i+j*global.gameBlockWidth];
        is2 =
blocksInfo[i+(j+1)*global.gameBlockWidth];
        is3 =
blocksInfo[i+(j+2)*global.gameBlockWidth];
        if( is1!= noone && is2!= noone && is3!=
noone ){
            if(( is2.gemsType ==
is1.gemsType )&&( is3.gemsType == is1.gemsType )){
                is1.alarm[0] = 1;
                is1.destory = true;
                is2.alarm[0] = 1;
                is2.destory = true;
                is3.alarm[0] = 1;
                is3.destory = true;
            }
        }
    }
}
```

Kode Sumber 4 ***Script checkingMatch***

```
if(mousePressed){
    x = global.goBackX;
    y = global.goBackY;
    global.goBackX = -1;
    global.goBackY = -1;
    global.pickupId = -1;
    global.holding = false;
    global.needToMatchCheck = true;
    mousePressed = false;
    depth = 100;
}
```

Kode Sumber 5 ***Script releaseHold***

```
if(!global.dropping && !global.holding && mousePressed==false){
    oldX = x;
    oldY = y;
    global.goBackX = x;
    global.goBackY = y;
    global.pickupId = uniqId;
    x=mouse_x;
    y=mouse_y;
    mousePressed = true;
    global.holding = true;
    show_debug_message("press mouse");
    depth = 50;
    alarm[2] = global.holdTime;
}
```

Kode Sumber 6 ***Script holdPress***

```
mousePressed = false;
gemsType = -1;
oldX = -1;
oldY = -1;
uniqId = -1;
havetoMoveX = -1;
havetoMoveY = -1;
havetoMove = false;
destory = false;
```

Kode Sumber 7 ***Script spriteObject create***

```
effect_create_above(ef_firework,x,y,1,65535);
alarm[1] = 30;
```

Kode Sumber 8 ***Script spriteObject Alarm 0***

```
instance_destroy();

global.point += global.pointGet;
```

Kode Sumber 9 ***Script spriteObject Alarm 1***

```
releaseHold();
```

Kode Sumber 10 ***Script spriteObject Alarm 2***

```
if( mousePressed ){
    x=mouse_x;
    y=mouse_y;
} else {
    if( global.holding ){
        if( !havetoMove &&
(distance_to_point(mouse_x,mouse_y) <= 0)

/*distance_square(x,y,mouse_x,mouse_y)<(global.blockSize/3)*(
global.blockSize/3)*/ ){
            var tempx = global.goBackX;
            var tempy = global.goBackY;

            if( havetoMove ){
                global.goBackX = havetoMoveX;
                global.goBackY = havetoMoveY;
            }else{
                global.goBackX = x;
                global.goBackY = y;
            }
            //x = tempx;
            //y = tempy;
            havetoMoveX = tempx;
            havetoMoveY = tempy;
            havetoMove = true;
            global.moving = true;
            depth = 90;
        }
    }
}

if( havetoMove ){
    if( x > havetoMoveX ) x=x-global.onceMoveStep;
    if( x < havetoMoveX ) x=x+global.onceMoveStep;

    if( y > havetoMoveY ) y=y-global.onceMoveStep;
    if( y < havetoMoveY ) y=y+global.onceMoveStep;

    if(x==havetoMoveX && y==havetoMoveY){
        depth = 100;
        havetoMove = false;
    }
}
```

Kode Sumber 11 ***Script spriteObject Step***

```
if( destory ) return 0;
holdPress();
```

Kode Sumber 12 ***Script spriteObject Left Pressed***

```
releaseHold();
```

Kode Sumber 13 ***Script spriteObject Left Released***

```
if( uniqId!=-1 && global.pickupId==uniqId ){

draw_sprite(spriteGoBack,0,global.goBackX,global.goBackY);
}
draw_self();

if( uniqId!=-1 && global.pickupId==uniqId ){

draw_sprite_part(spriteRelease,0,0,0,global.spriteReleaseW
idth,
                    global.spriteReleaseHeight,x,y);

draw_sprite_part(spriteRelease,1,0,0,((global.holdTime-
(global.holdTime-
alarm[2]))*global.spriteReleaseWidth/global.holdTime),
                    global.spriteReleaseHeight,x,y);
}
```

Kode Sumber 14 ***Script spriteObject Create***

```
if(global.currentLevel==1){
    countH = 4;
    global.genBlockTime = 360;
    alarm[0] = global.genBlockTime;
}

for(i=0;i<countH;i++){
    for(j=0;j<global.gameBlockWidth;j++){
        var ins =
instance_create(global.gameStartX+j*global.blockSize,gl
obal.gameStartY-i*global.blockSize,spriteObject);
        randomizeSprite(global.currentLevel,ins);
    }
}
```

Kode Sumber 15 ***Script controlObject Create***

```
global.genBlock = true;
```

Kode Sumber 16 ***Script controlObject Alarm 0***

```
if(global.gameover) return 0;

if( !global.holding && !global.moving && !global.dropping ){
    var is1 = noone;
    var i,j;
    for(j=0;j<global.gameBlockHeight;j++){
        for(i=0;i<global.gameBlockWidth;i++){
            is1 = noone;

if( !position_empty(global.gameStartX+i*global.blockSize,global
.gameStartY-j*global.blockSize) ){
                is1 =
instance_position(global.gameStartX+i*global.blockSize,global.g
ameStartY-j*global.blockSize,spriteObject);
            }
            blocksInfo[i+j*global.gameBlockWidth]=is1;
        }
    }

    if( global.needToMatchCheck ){
        global.needToMatchCheck = false;
        checkingMatch(blocksInfo);
    }

    if( !global.dropping && global.genBlock ){
        var j;
        for(j=0;j<global.gameBlockWidth;j++){

if( position_empty(global.gameStartX+(j*global.blockSize),globa
l.gameStartY-((global.gameBlockHeight)*global.blockSize)) ){
                var ins =
instance_create(global.gameStartX+(j*global.blockSize),
                                        global.gameStartY-
((global.gameBlockHeight)*global.blockSize),
                                        spriteObject);
                randomizeSprite(global.currentLevel,ins);
            }else{
                instance_create(room_width/2, room_height/2,
gameoverObject);
                global.gameover = true;
                return 0;
            }
        }
        global.genBlock = false;
        alarm[0] = global.genBlockTime;
    }
```

```
    for (i = 0; i < instance_number(spriteObject); i +=
1)
    {
        var is1 = noone;
        is1 = instance_find(spriteObject,i);
        if(is1!=noone){
            with(is1){
                if(place_free(x,y+global.onceMoveStep)
and (y<global.gameStartY)){
                    global.dropping = true;
                    break;
                }
            }
        }
    }

}

if( global.dropping ){
    var i;
    var is1 = noone;
    var drop = false;
    for (i = 0; i < instance_number(spriteObject); i +=
1)
    {
        is1 = instance_find(spriteObject,i);
        if(is1!=noone){
            with(is1){
                if(place_free(x,y+global.onceMoveStep)
and (y<global.gameStartY)){
                    drop = true;
                    y+=global.onceMoveStep;
                }
            }
        }
    }
    if( !drop ) {
        global.dropping = false;
        global.needToMatchCheck = true;
    }
}
```

```
if( global.moving ){
    var i;
    var is1 = noone;
    var move = false;
    for (i = 0; i < instance_number(spriteObject); i +=
1)
    {
        is1 = instance_find(spriteObject,i);
        if(is1!=noone){
            with(is1){
                if(havetoMove){
                    move = true;
                    break;
                }
            }
        }
    }
    if( !move ) {
        global.moving = false;
        global.needToMatchCheck = true;
    }
}
```

Kode Sumber 17 ***Script controlObject Step***

```
draw_text(10,10,global.dropping);
draw_text(10,20,global.holding);
draw_text(10,30,global.moving);
draw_set_color(c_white);
draw_text(220, 20, string_lettersdigits("Score: " +
string(global.point)));
```

Kode Sumber 18 ***Script controlObject Draw***

BIODATA PENULIS

Muhammad Husain Fuad Dzulfikri dilahirkan pada tanggal 18 Maret 1994 di Kediri. Pada tahun 2012,sesudah lulus dari SMAN 1 Kediri melanjutkan menimba ilmu di jurusan Teknik Informatika Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember Surabaya.

Memiliki pengalaman organisasi sebagai Staf Departemen Sosial Masyarakat Fakultas Teknologi Informasi 2014-2015, Staf Departemen Hubungan Masyarakat Keluarga Muslim Informatika 2014-2015, Selain itu, juga memiliki pengalaman kepanitiaan sebagai staf Pertrans Schematics 2013.