PENCARIAN RUTE TERPENDEK DALAM DUNIA 3 DIMENSI BERDASARKAN ALGORITMA DIJKSTRA

Ahmad Hoirul Basori[1] Andi Tenriawaru[2]

[1]Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
[2]Jurusan Matematika , Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Haluoleo
Email: hoirul@its-sby.edu, ilmirahman@yahoo.com

ABSTRACT
Virtual reality is one of the favorite application that very liked by peoples. Vitual reality also used in many games application in order to show the imitation of real world. Recently, virtual reality still imperfect due to shortest path problem. It need more accuration enhancement to get a very good refelection of the real world. This paper intends to discuss how to enhance the accuration of virtual reality shortest path tracking by utilizing Dijkstra algorithm. The built up prototype has three abilities: visualization, simulation for campus traffic, and exploration of campus world in 3D ways.

Keywords: *3D, virtual reality, dijkstra*

ABSTRAK
Virtual reality merupakan salah satu aplikasi yang sangat digemari oleh masyarakat akhir-akhir ini. Virtual reality banyak digunakan dalam game-game untuk menggambarkan suatu situasi dalam dunia nyata mendekati aslinya. Selain itu pencarian rute terpendek pada dunia 3 dimensi masih mempunyai kekurangan terutama dari sisi akurasi. Oleh karena itu paper ini akan membahas tentang pemanfaatan algoritma dijkstra untuk melakukan pencarian rute terpendek pada dunia 3 dimensi. Prototype pada penelitian ini mempunyai 3 kemampuan: visualisasi, simulasi tentang jalan-jalan di kampus, serta proses penjelajahan kampus melalui dunia 3 dimensi.

Kata Kunci: *3D, virtual reality, dijkstra*

Dunia 3 dimensi bukan hanya menjadi angan-angan di zaman sekarang ini, tetapi dengan munculnya prosesor baru dan *graphics card* baru yang lebih canggih mendorong pertumbuhan aplikasi 3D menjadi lebih mudah dan cepat.

Dalam kehidupan nyata, petunjuk jalan di suatu daerah sangatlah penting, terutama jika daerah sangat luas dan kompleks. Akan tetapi, penelusuran suatu daerah secara nyata memakan waktu yang cukup lama dan tenaga dan besar [1]. Dengan keberadaan dunia 3D kita dimungkinkan berfantasi seolah-olah sudah berada di area tersebut dan menghapalkan daerahnya dalam bentuk simulasi di komputer [2].

Dalam penelitian ini akan dibuat suatu program yang mempresentasikan suatu dunia 3 dimensi kampus ITS. Dalam hal ini, setiap posisi gedungnya ditempatkan sesuai dengan peta kampus ITS sesungguhnya.

Selain itu, algoritma Dijkstra yang sudah banyak digunakan dalam pencarian rute terpendek masih kurang sekali penerapannya pada dunia 3 dimensi. Hal ini dikarenakan penerapan Dijkstra pada dunia 3 dimensi harus mempertimbangkan berbagai aspek, seperti pengaturan kamera, *setting* posisi, dan skala dunia (*world scale*).

Kontribusi penelitian ini adalah penerapan algoritma Dijkstra pada area dunia 3 dimensi dengan menggunakan beberapa fitur koordinat dan penggunaan kamera pada dunia 3 dimensi sebagai elemen pada algoritma Dijkstra.

3D TOOLS

Untuk menghasilkan tampilan grafis 3D yang realistis maka tentu saja objek-objek tidak akan digambar melalui *coding*. Apabila semua objek dikerjakan melalui *coding* tentu saja akan memakan waktu dan tampilannya pun tampak tidak realistis. Namun dengan bantuan *tool-tool* untuk 3D-*modeling* serta *tool* animasi, hal ini dapat diatasi.

Tool-tool ini menawarkan fitur-fitur, seperti *multiple windows* agar objek yang didesain bisa dilihat dari tiap dimensi. Salah satu tool untuk 3D-*modeling* yang bagus adalah Google Sketchup, 3D Studio Max, serta Blender.

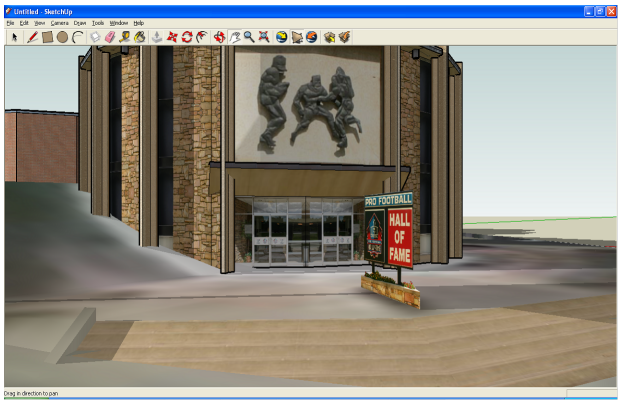
Google Sketchup adalah *tool* untuk membuat model dan desain 3D serta dapat juga digunakan untuk membuat animasi baik yang sederhana maupun yang kompleks. *Tool* ini memberikan kontribusi yang cukup besar pada pemodelan 3D, penggambaran arsitektural, desain interior, dan animasi.

Tool ini mampu menghasilkan efek-efek realistis seperti kabut, *lighting* (pencahayaan), dan api yang berkobar. Disediakan pula Material Editor untuk menentukan warna serta *texture mapping*.

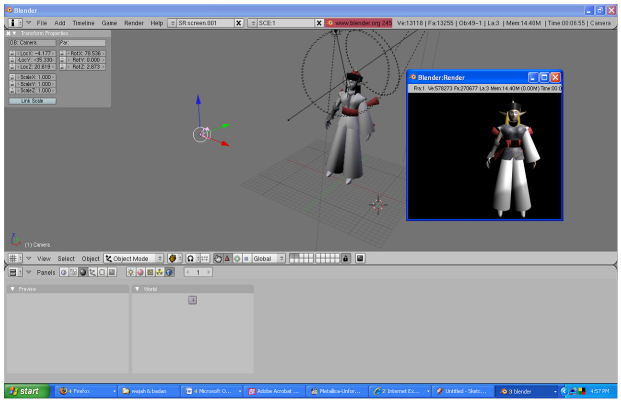
Google Sketchup juga dilengkapi dengan fasilitas pengaturan kamera. Kamera ini dapat diposisikan, diarahkan, dan dikendalikan. Dalam satu scene 3D, bisa terdiri atas lebih dari satu kamera. Di sini bisa ditentukan mana kamera yang harus aktif, mana yang tidak. Kamera juga bisa diatur tingkat fokusnya, jarak pandangnya, dan sebagainya.

Pada Google Sketchup terdapat pula *plugin* seperti *polygon counter* yang dapat dipergunakan untuk menghitung jumlah poligon untuk keperluan *modeling*. Kemudian pada saat melakukan pemodelan, pemodel bisa mendesain sambil memperhatikan jumlah poligon yang sudah terpakai.

Google Sketchup [3] dan Blender [4] merupakan *tools* untuk pemodelan 3D yang gratis dan mudah dipakai. Oleh karena itu banyak programmer yang menggunakan *tools* ini untuk membangun suatu objek 3 dimensi (lihat Gambar 1 dan Gambar 2).



Gambar 1: Hasil objek 3D dengan Google Sketchup



Gambar 2: Hasil objek 3D dengan Blender

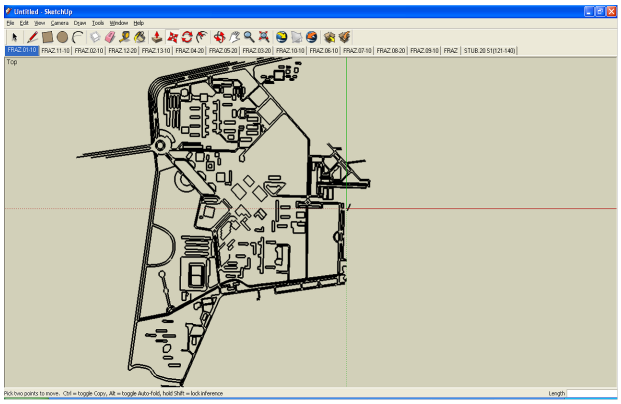
PEMODELAN OBJEK 3D

Dengan hadirnya beberapa *game engine*, keinginan untuk membangun aplikasi 3D pada PC menjadi kenyataan. Dengan 3D, konsep realistik suatu grafis menjadi terwujudkan. Meskipun demikian, masih ada beberapa hal yang harus diperhatikan mengingat kemampuan perangkat bergerak yang terbatas.

Sebagaimana yang sudah diketahui, seiring dengan meningkatnya *processing power* serta kemampuan *video card*, jumlah data yang diproses dalam gambar 3D meningkat setiap hari setiap tahunnya, untuk menghasilkan tingkat kedetailan gambar yang tinggi. Namun untuk menghasilkan tingkat kedetailan seperti di atas diperlukan banyak kalkulasi matematis untuk menciptakan *image* per detiknya. Ka-rena waktu dan pemrosesan terbatas, maka semua aspek yang membentuk scene 3D seperti geometri, animasi, pencahayaan, dan material harus dioptimasikan.

Istilah *low poly* dalam 3D modeling berarti membangun objek dengan jumlah poligon yang rendah. Tidak ada ukuran yang pasti seberapa rendah nilai jumlah yang harus digunakan sebab jumlah poligon bergantung pada platform yang dipakai (PC, konsol, serta perangkat bergerak).

Dalam dunia 3D, semua objek dibentuk dari segitiga. Jumlah poligon yang terpakai berarti jumlah poligon yang sudah dipergunakan untuk membentuk objek tersebut. Dalam 3dsmax, objek dapat berupa Editable Mesh (mesh terdiri dari segitiga, sehingga dimanipulasi pada level segitiga), dan Editable Poly (mesh terdiri dari kotak, yang tersusun dari 2 segitiga). Sebagai contoh, objek kubus terdiri dari 6 sisi (kotak). Jika diconvert ke Editable Mesh maka kubus tersebut terdiri dari 12 poligon. Sebab satu kotak terbentuk dari 2 segitiga yang digabungkan.



Gambar 3: Gambar peta kampus ITS dalam 2D

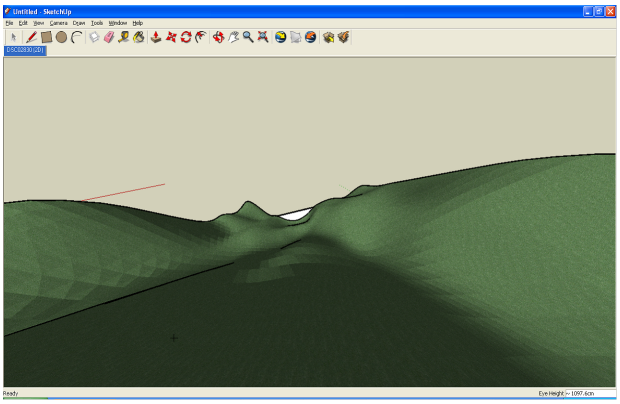
Pada waktu melakukan pemodelan, harus selalu diperhatikan jumlah poligon yang terpakai. Untuk melakukan hal ini, pemodel dapat menggunakan *Polygon Counter*, salah satu *tool* yang terdapat pada 3dsmax. Namun sebelum menggunakan *Polygon Counter*, semua objek hendaknya terlebih dahulu dikonversi ke *Editable Mesh*, sehingga *Polygon Counter* memberikan report yang benar mengenai jumlah poligon yang terpakai. Jika tidak dilakukan maka *Polygon Counter* akan memberikan report yang salah yaitu jumlah poligon dihitung lebih kecil dari sebenarnya disebabkan objek berupa *Editable Poly* dan bukan *Editable Mesh*.

Dalam dunia nyata, terdapat jumlah poligon yang tak terhitung jumlahnya. Namun dalam aplikasi 3D, semua objek terbentuk dari segitiga. Hal lain yang perlu diketahui adalah pengaturan kamera. Pada Google Sketchup, terdapat dua jenis kamera, yaitu *free camera* dan *target camera*. Pada *free camera*, arah sorot kamera diatur secara manual oleh pemodel. Namun pada *target camera*, arah sorot kamera, terpaku pada satu objek sehingga jika objek tersebut bergerak, arah sorot kamera akan selalu berubah sesuai posisi objek yang disorot.

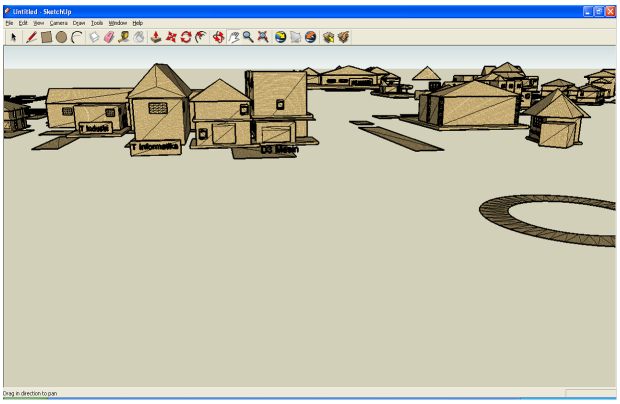
PEMBUATAN PETA 3 DIMENSI

Objek-objek 3 dimensi yang dapat digunakan dalam pemrograman berbasis 3D dapat dibuat menggunakan berbagai macam program desain 3D seperti Google SketchUp, 3D Studio Max, Maya, Blender, dan sebagainya. Dalam penelitian ini akan dibahas proses pembuatan peta 3 dimensi dengan menggunakan bahan mentah data vektor kontur dan sketsa dari peta yang sudah dibuat di Autocad (lihat Gambar 3). Proses pemodelan dilakukan dengan menggunakan program desain 3 dimensi Blender dan Google SketchUp.

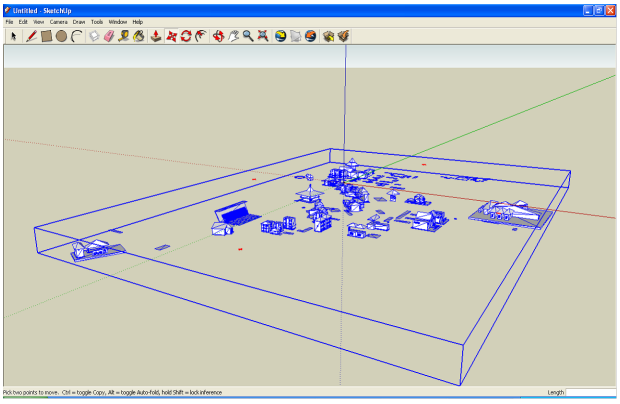
Langkah pertama dalam pembuatan peta 3 dimensi adalah *import* data vektor kontur 3 dimensi kedalam aplikasi Google SketchUp. Data kontur tersebut kemudian dibuat menjadi model 3 dimensi sehingga dapat menunjukkan ben-



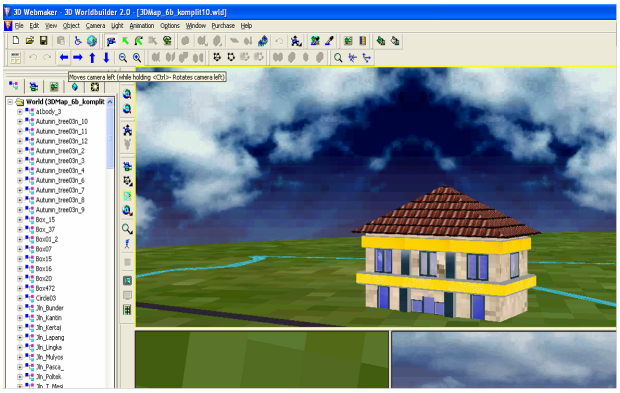
Gambar 4: Contoh miring



Gambar 6: Desain 3D kampus ITS dengan tekstur



Gambar 5: Model peta kampus ITS dilengkapi bangunan



Gambar 7: Desain 3D kampus ITS dalam 3D Web Maker

tuk sebuah daratan seperti yang disebut *terrain* (lihat Gambar 4). Pada saat pembangunan dunia 3D kampus yang sesungguhnya objek 3D dari masing-masing bangunan harus diletakkan sesuai dengan posisi yang ada pada peta dan dicocokkan dengan kondisi aslinya. Oleh karena itu letak dan koordinat dari masing-masing bangunan harus diketahui terlebih dahulu.

Langkah kedua adalah menambahkan bangunan pada peta dua dimensi sehingga terlihat gambar-gambar bangunan seperti pada Gambar 5. Setelah bangunan ditempatkan sesuai dengan posisinya, bangunan tersebut harus diberi tekstur agar kelihatan lebih nyata dan lebih bagus. Kemudian masing-masing bangunan diberi nama sesuai dengan nama aslinya seperti yang terlihat pada bangunan 3 Dimensi di Gambar 6 (catatan: sudah terdapat label T.Informatika, T.Industri, serta D3 Me-sin).

Langkah ketiga adalah melakukan *export* model 3 dimensi dengan menggunakan Google SketchUp agar menjadi file berformat DirectX *desktop* dengan ekstensi *.x*. Setelah itu file harus diexport lagi kedalam bentuk *.wld* agar terbaca pada *game engine* 3dstate.

PROSES PEMUATAN OBJEK 3D KE DALAM GAME ENGINE

Setelah objek selesai dimodelkan, maka yang harus dilakukan adalah memuat objek 3D ke dalam aplikasi. Ada beberapa cara untuk memuat objek 3D yaitu menggunakan *framework direct* X, 3dstate engine, *irrlicht game engine*, dan lain lain. Pada penelitian ini, *game engine* yang digunakan adalah 3dstate yang menggunakan ekstensi *.wld* dalam melakukan *loading object* 3D-nya.

PEMBUATAN *VIRTUAL REALITY* PADA *DESKTOP ENVIRONMENT*

Seperti yang sudah diutarakan di bagian sebelumnya, penelitian ini menggunakan *game engine* 3dstate untuk proses *loading object* 3D-nya.

Tahap pertama yang dilakukan adalah melakukan proses konversi sehingga model yang dibuat dapat menjadi *.wld*. Setelah itu model bisa dikustomisasi dengan menggunakan 3D web maker seperti yang terlihat pada Gambar 7 dan Gambar 8. Setelah itu buatlah *form* di C# dan masukkan *code* yang tertulis di Gambar 9 pada *event form load*.

Algoritma Dijkstra yang ditemukan oleh Edsger Dijkstra menggunakan prinsip *greedy* [5]. Prinsip tersebut memecahkan masalah lintasan terpendek untuk sebuah graf berarah dengan bobot sisi yang tidak negatif. Implementasi dapat dilihat pada potongan *code* di Gambar 10. Untuk bisa membuat simulasi yang dapat mencari rute terpendek, penelitian ini menggunakan algoritma Dijsktra dalam menyelesaikan proses pencarian *shortest path*.

Gambar 8: Desain 3D kampus ITS dan aksesoris pepohonan

```
   private void Form2_Load(object sender, System.EventArgs e)
   {
       if(m_AlreadyDone) return;
       m_AlreadyDone = true;
       string Path = Application.ExecutablePath;
       Path = System.IO.Path.GetDirectoryName(Path);
       if(!MLoadWorld("3DMap_6b_komplit10.wld", Path)) return;

       MInitCamera();
       MInitialize();
       MInitHardware();

       _3D.STATE_engine_hide_log_window();
       true;
   }
```

Gambar 9: Kode program C# pada *event form load*

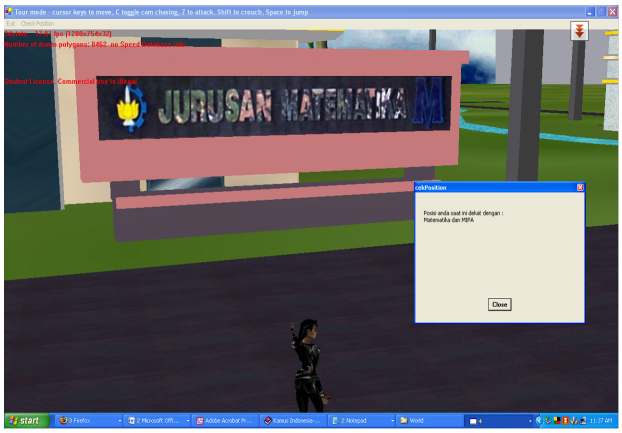
```
public WindowsApplication1.djikstraResult getShortest(int start, int finish)
{
       int i = 0;
       int j;
       int permanent;
       int min;

       WindowsApplication1.djikstraResult ruteBaru = new djikstraResult();

       Boolean ketemu = false;
       Boolean found = false;

       Label.label.AddlabelRow(i, start, 0, start, true);

       i = i + 1;
       permanent = start;
       while(!ketemu){
          for(j = 0;
             j<System.Convert.ToInt16(System.Math.Sqrt(tabel.Length) - 1);
             j++)
          {
             if(tabel[permanent, j] > 0 && this.checkstatus(j)='a')
             {
                Label.label.AddlabelRow
                     (i, permanent,
                    Label.label[this.findnode(permanent)].labelJarak
                    + tabel[permanent, j])
                i = i + 1;
             }
             else if(tabel[permanent, j] > 0 && this.checkstatus(j)='b')
             ... ...
          }
       }
}
```

Gambar 10: Potongan *source code* untuk implementasi algoritma Dijkstra

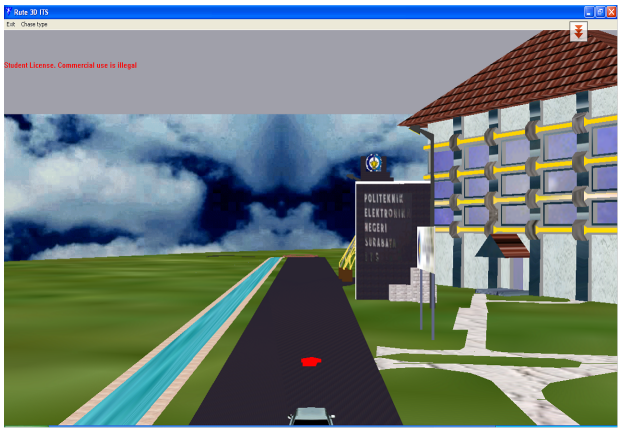
```
#region Rektorat
tabel[45,46] = 2;
tabel[46,45] = 2;
#endregion Rektorat

#region BAAK
tabel[56,57] = 2;
tabel[57,56] = 2;
#endregion BAAK

#region T Informatika
tabel[474,475] = 2;
tabel[475,474] = 2;
#endregion T Informatika

#region D3 Mesin
tabel[482,483] = 2;
tabel[483,482] = 2;
#endregion D3 Mesin

#region T Elektro
tabel[582,583] = 2;
tabel[583,582] = 2;
#endregion T Elektro
```

Gambar 11: Koordinat pada desain 3D Kampus ITS

(a) Tampilan saat *loading* 3D Model secara keseluruhan.



(b) Proses penelusuran 3D mulai dari pintu depan sampai di depan jurusan Matematika.



(c) Proses simulasi dari pintu depan ITS menuju ke T.Perkapalan.

(d) Proses simulasi selesai karena titik akhir T.Perkapalan sudah dicapai.

Gambar 12: Contoh-contoh tampilan saat uji coba

Sebelum menggunakan Dijsktra untuk pencarian *path*, terlebih dahulu kita harus menentukan koordinat untuk masing-masing bangunan yang ada di 3D model. Contoh koordinat yang digunakan pada uji coba ditunjukkan pada Gambar 11). Setelah menempatkan semua *object* 3D posisi koordinat yang sesuai, maka langkah selanjutnya adalah melakukan *load* terhadap *object* secara keseluruhan, sehingga tampilan awal seperti yang terlihat pada Gambar 12(a).

UJI COBA

Uji coba terhadap aplikasi ini dilakukan pada PC dengan skenario uji coba sebagai berikut Uji Coba Tour Kampus dan Uji Coba Simulasi Guide Kampus ITS (lihat Gambar 12).

Uji Coba Tour Kampus menekankan pada kebebasan user untuk menjelajahi kampus secara 3Dimensi tanpa harus terpaku pada titik awal dan titik akhir. Proses penelusuran dapat menggunakan *key* **Arrow Up Down** untuk gerakan maju mundur, serta *key* **Left Right** untuk bergerak ke kanan atau kiri. Aplikasi juga dilengkapi dengan fasilitas untuk mengecek posisi saat paling dekat dengan bangunan apa. Sebagai contoh pada Gambar 12(b), penelusuran sudah sampai di dekat jurusan Matematika-MIPA sehingga kita proses cek posisi maka program akan menampilkan posisi terdekat adalah Matematika-MIPA.

Sedangkan pada Uji Coba Simulasi Guide Kampus ITS menekankan pada proses simulasi suatu rute dari titik awal tertentu ke titik akhir yang menjadi tujuan. Simulasi ini menggunakan *object* mobil yang akan bergerak mengikuti *track* yang sudah ada dengan mencari rute terpendek berdasarkan algoritma Dijkstra.

Pada Gambar 12(c) yang diujicobakan sebagai titik awal adalah **Pintu depan (pos Saptam Bundaran ITS)** menuju ke **Teknik Perkapalan**. Aplikasi ini akan memeriksa rute manakah yang terpendek dengan membandingkan jalur-jalur yang akan dilalui. Dengan demikian, aplikasi akan memilih **Jalur 2** sebagai jalur terpendek. Proses simulasi dapat dilihat pada Gambar 12(c) dan Gambar 12(d).

Contoh jalur yang dilalui adalah Jalur 1: pintu depan → taman alumni → kantin ITS → Teknik Elektro → Teknik Kimia → Teknik Perkapalan. Sedangkan jalur lain yang bisa dilewati adalah Pintu Depan → Poltek Elektronika → Poltek Perkapalan → Teknik Perkapalan.

SIMPULAN

Simpulan yang dapat diambil dari beberapa uji coba yang telah dilakukan, antara lain:

Proses penelusuran dengan mode TOUR masih belum bisa mencerminkan jarak aslinya. Hal ini dikarenakan proses skalanya masih belum disesuaikan dengan skala yang sebenarnya dari kampus ITS, tetapi posisi dari setiap gedung berada pada posisi yang sama dengan keadaan dunia nyata kampus ITS.

Mode simulasi sudah bisa berjalan sesuai dengan algoritma Dijkstra yaitu mencari rute terpendek. Akan tetapi, ketepatannya masih kurang teliti karena penentuan letak masing-masing bangunan masih menggunakan range koordinat.

Penggunaan skala pada dunia 3 dimensi, sehingga jarak tempuh bisa diperkirakan mendekati aslinya. Selain itu penggunaan range yang lebih presisi untuk setiap bangunan agar proses simulasi dengan menggunakan Dijkstra juga bisa meningkatkan hasil dengan baik.

DAFTAR PUSTAKA

[1] Moser, R.: *A Fantasy Adventure Game as a Learning Environment: Why Learning to Program is So Difficult and What Can be Done About It*. SIGCSE Bull. **29**(3) (1997) 114–116

[2] Nakamura, R., Tori, R., Bernardes JR., J., Bianchini, R., Jacober, E.: *A Practical Study on the Usage of a Commercial Game Engine for the development of Educational Games*. In: 2nd Games and Digital Entertainment Workshop, Brazilian Computer Society (2003)

[3] *Google SketchUp*: `http://sketchup.google.com/`.

[4] Blender: `http://www.blender.org/`.

[5] Dijkstra, E.W.: *A Note on Two Problems in Connexion with Graphs*. Numerische Mathematik **1** (1959) 269–271