SKRIPSI

PEMANFAATAN SMARTPHONE SEBAGAI PENGENDALI PERMAINAN BERBASIS WEB

Priambodo Pangestu

NPM: **2013730055**

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN

«tahun»

UNDERGRADUATE THESIS

«JUDUL BAHASA INGGRIS»

Priambodo Pangestu

NPM: 2013730055

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY

«tahun»

LEMBAR PENGESAHAN

PEMANFAATAN SMARTPHONE SEBAGAI PENGENDALI PERMAINAN BERBASIS WEB

Priambodo Pangestu

NPM: 2013730055

Bandung, «tanggal» «bulan» «tahun»

Menyetujui,

Pembimbing Utama	Pembimbing Pendamping
«pembimbing utama/1»	«pembimbing pendamping/2»

Ketua Tim Penguji	Anggota Tim Penguji
«penguji 1»	«penguji 2»

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

PEMANFAATAN SMARTPHONE SEBAGAI PENGENDALI PERMAINAN BERBASIS WEB

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal «tanggal» «bulan» «tahun»

Meterai
Rp. 6000

Priambodo Pangestu
NPM: 2013730055

ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Kata-kata kunci: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»
Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Keywords: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»

«kepada siapa anda mempersembahkan skripsi ini. . . ?»

KATA PENGANTAR

«Tuliskan kata pengantar dari anda di sini . . . »
Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Bandung, «bulan» «tahun»

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Latar Belakang

WebSockets adalah teknologi yang memungkinkan *web browser* pengguna dan *web server* membuka sesi komunikasi interaktif satu sama lain. Teknologi *WebSockets* didesain untuk diimplementasikan pada *web browser* dan *web server*, tetapi dapat juga digunakan oleh setiap aplikasi *client* maupun *server*. *WebSockets* memiliki standar yang menyediakan cara agar *web server* dapat mengirim konten ke *web browser* tanpa diminta oleh *client*, dan memungkinkan agar pesan dikirimkan berulang-ulang dengan tetap menjaga koneksi yang terbuka. Oleh karena itu, protokol *WebSockets* memungkinkan interaksi antara *web browser* dan *web server* dengan *overhead* yang rendah, dan juga memfasilitasi transfer data *realtime* dari *server* maupun menuju *server*.

Salah satu teknologi yang memanfaatkan protokol *WebSockets* adalah *Socket.io*. Teknologi ini memungkinkan untuk melakukan komunikasi secara *realtime*, dan dua arah antara *client* dan *server*. Socket.io memiliki dua bagian: *client-side library* yang berjalan didalam *web browser*, dan *server-side library* yang berjalan pada *Node.js*. *Socket.io* memiliki fitur-fitur yang beragam, seperti melakukan broadcast ke beberapa *sockets*, dan menyimpan data yang berhubungan dengan masing-masing *client*. Teknologi ini sangat berguna untuk membantu membangun sebuah aplikasi yang membutuhkan koneksi *realtime* seperti dalam aplikasi *chatting* maupun *game*.

Untuk memanfaatkan protokol *WebSockets* dalam membangun aplikasi permainan, akan dibutuhkan beberapa teknologi yang dapat membantu pembangunan aplikasinya. Salah satu teknologi tersebut yaitu *Canvas API*. Teknologi ini merupakan bagian dari *HTML5 element* yang dapat digunakan untuk menggambar suatu grafis melalui *JavaScript* secara *on the fly*. *Canvas API* dapat juga digunakan untuk membuat komposisi foto, membuat animasi, dan membuat *real-time video processing* atau *rendering*. Oleh karena itu, fungsi-fungsi yang ada pada *Canvas API* akan membantu pembangunan aplikasi permainan terutama pada bagian pengembangan grafis pada aplikasinya.

Teknologi lain yang dapat membantu membangun aplikasi permainan dalam memanfaatkan protokol *WebSockets* yaitu *Node.js*. Teknologi ini merupakan sebuah *platform* yang didesain untuk mengembangkan aplikasi berbasis web pada bagian *web server*. *Node.js* ditulis dalam sintaks bahasa pemrograman *JavaScript* dan menggunakan *V8* yang merupakan *engine JavaScript* milik perusahaan *Google* untuk mengeksekusi *JavaScript* pada *web server*. *Node.js* memiliki sifat *non-blocking*, yang berarti *Node.js* tidak akan menunggu untuk mengerjakan *request* selanjutnya. *Node.js* pun sangat cepat dalam mengeksekusi suatu kode karena menggunakan *engine JavaScript V8*. Fitur-fitur yang dimiliki oleh *Node.js* akan sangat membantu untuk membangun aplikasi permainan yang membutuhkan koneksi *real-time*.

Pada skripsi ini, akan dibuat sebuah aplikasi permainan yang memanfaatkan protokol *WebSockets*, dimana dalam penggunaan protokol tersebut akan dibantu dengan teknologi *Socket.io*. Selain itu, aplikasi yang dibuat akan memanfaatkan *personal computer (PC)* dan *smartphone* untuk pengembangan aplikasinya. Para pemain akan mengkoneksikan *smartphone* miliknya pada suatu *PC*, dimana *smartphone* tersebut akan berfungsi sebagai *controller* untuk memainkan permainannya. Oleh karena itu, protokol *WebSockets* akan digunakan sebagai koneksi antara *smartphone* dan *PC*

dalam aplikasi permainan yang akan dibangun. Aplikasi permainan akan menggunakan teknologi berbasis web, sehingga untuk memainkannya, *client* bisa mengakses melalui *web browser* tanpa harus berada di satu jaringan lokal yang sama.

1.2 Rumusan Masalah

1. Bagaimana membangun aplikasi permainan berbasis web dengan memanfaatkan protokol *WebSockets* untuk penggunaan *smartphone* sebagai pengendali permainan berbasis web ?
2. Berapa *latency* yang dihasilkan berdasarkan penggunaan protokol *WebSockets* ?

1.3 Tujuan

1. Mengetahui cara membangun aplikasi permainan berbasis web dengan memanfaatkan protokol *WebSockets* untuk penggunaan *smartphone* sebagai pengendali permainan berbasis web.
2. Mengetahui jumlah *latency* yang dihasilkan berdasarkan pemanfaatan protokol *WebSockets*.

1.4 Batasan Masalah

Batasan masalah yang dibuat terkait dengan pengerjaan skripsi ini adalah sebagai berikut:

1. Aplikasi permainan yang dibuat merupakan permainan *multiplayer* yang hanya bisa dimainkan oleh dua orang saja.

1.5 Metodologi

Metodologi yang dilakukan dalam pengerjaan skripsi ini adalah sebagai berikut:

1. Studi literatur mengenai :
   - *WebSockets* yang akan digunakan untuk koneksi antara *smartphone* dan *PC*.
   - *Socket.io* sebagai teknologi yang akan menggunakan *WebSockets* dalam pembangunan aplikasi.
   - *Canvas API* yang akan digunakan untuk antarmuka permainan.
   - *Node.js* sebagai *web server* dalam pembangunan aplikasi.
   - *Express.js* sebagai *Node.js framework* yang akan digunakan untuk mengatur penyimpanan data dalam *Node.js*
2. Menganalisis aplikasi sejenis.
3. Merancang antarmuka permainan pada *PC* dan *smartphone*. Antarmuka pada *PC* akan berbeda dengan yang ada di *smartphone*, karena *smartphone* akan bekerja sebagai *controller* dan *PC* akan bekerja sebagai *console*.
4. Menyusun cara bermain aplikasi permainan yang dibangun.
5. Mengimplementasi program aplikasi permainan berbasis web.
6. Menganalisis *latency* yang dihasilkan pada aplikasi.
7. Melakukan eksperimen dan pengujian yang melibatkan responden.

1.6 Sistematika Pembahasan

Setiap bab dalam skripsi ini memiliki sistematika penulisan yang dijelaskan kedalam poin-poin sebagai berikut:

1. Bab 1 : Pendahuluan
   Membahas mengenai gambaran umum penelitian ini. Berisi tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metode penelitian, dan sistematika penulisan.

2. Bab 2 : Dasar Teori
   Membahas mengenai teori-teori yang mendukung berjalannya penelitian ini. Berisi tentang *WebSockets*, *Socket.io*, *Node.js*, *Express.js*, dan *Canvas API*.

3. Bab 3 : Analisis
   Membahas mengenai analisa masalah.

4. Bab 4 : Perancangan
   Membahas mengenai perancangan yang dilakukan sebelum melakukan tahapan implementasi.

5. Bab 5 : Implementasi dan Pengujian
   Membahas mengenai implementasi dan pengujian yang telah dilakukan.

6. Bab 6 : Kesimpulan dan Saran
   Membahas hasil kesimpulan dari keseluruhan penelitian ini dan saran-saran yang dapat diberikan untuk penelitian berikutnya.

BAB 2

LANDASAN TEORI

Pada bab ini akan dijelaskan landasan teori mengenai *WebSockets*, *Socket.io*, *Node.js*, *Express.js*, dan *Canvas API*.

2.1 WebSockets

WebSockets merupakan *Application Programming Interface (API)* yang memungkinkan sebuah aplikasi membuka sesi komunikasi interaktif antara *browser* pengguna dan *server*. Dengan API ini, pengguna dapat mengirim pesan ke *server* dan menerima respon tanpa harus melakukan *polling* pada *server* terlebih dahulu.

Subbab-subbab berikut menjelaskan beberapa kelas dari *WebSockets*.

2.1.1 WebSocket

Sebuah objek dari kelas *WebSocket* menyediakan *API* untuk membuat dan mengelola koneksi *WebSocket* ke *server*, dan juga untuk mengirim dan menerima data pada koneksi. Konstruktor pada kelas *WebSocket* menerima satu parameter wajib dan satu parameter pilihan:

WebSocket WebSocket(in DOMString url, in optional DOMString protocols);

url, merupakan parameter wajib yang menunjukan URL mana yang akan direspon oleh *WebSocket server*.

protocols, merupakan parameter pilihan yang dapat berupa satu protokol dengan tipe *string*, atau beberapa protokol dengan tipe *array of strings*. Apabila protokol spesifik tidak dimasukan pada parameter, maka akan diasumsikan sebagai *string* kosong.

Beberapa atribut yang dimiliki oleh kelas *WebSocket* yaitu sebagai berikut :

- **readyState**
  Atribut ini menunjukan status dari sebuah koneksi.
- **onclose**
  Atribut ini merupakan *event listener* yang akan dipanggil pada saat status koneksi *WebSocket* berubah menjadi CLOSED.
- **onerror**
  Atribut ini merupakan *event listener* yang akan dipanggil apabila terjadi *error*.
- **onmessage**
  Atribut ini merupakan *event listener* yang akan dipanggil apabila pesan dari server telah diterima.
- **onopen**
  Atribut ini merupakan *event listener* yang akan dipanggil pada saat status koneksi *WebSocket* berubah menjadi OPEN.

Kelas *WebSocket* memiliki dua buah *method*, yaitu :

- close()
  Berfungsi untuk menutup koneksi *WebSocket* atau menghentikan apabila sedang ada proses koneksi. *Method* ini memiliki tipe kembalian *void*, sehingga tidak akan mengembalikan apapun.

- send()
  Berfungsi untuk mengirim data ke *server* melalui koneksi *WebSocket*. *Method* ini memiliki parameter **data** yang merupakan sebuah *string* yang akan dikirimkan ke *server*.

2.1.2 CloseEvent

CloseEvent akan dikirim ke *client* menggunakan protokol *WebSockets* ketika koneksi sudah tertutup. *Constructor* dari kelas ini yaitu :

CloseEvent()

Properti yang dimiliki oleh kelas ini yaitu :

- **CloseEvent.code**
  Mengembalikan sebuah kode untuk menutup koneksi yang dikirimkan oleh *server*.

- **CloseEvent.reason**
  Mengembalikan alasan dari koneksi yang telah ditutup oleh *server*

- **CloseEvent.wasClean**
  Mengembalikan *boolean* yang mengindikasi apakah sebuah koneksi sudah tertutup sepenuhnya atau belum.

2.1.3 MessageEvent

Kelas ini merepresentasikan pesan yang diterima oleh suatu objek tertentu. *Constructor* dari kelas ini yaitu :

MessageEvent()

Beberapa properti yang dimiliki oleh kelas ini yaitu :

- **MessageEvent.data**
  Merupakan data yang telah dikirimkan oleh pengirim.

- **MessageEvent.lastEventId**
  Merepresentasikan *ID* yang unik untuk sebuah *Event*.

2.1.4 WebSocket Events

WebSockets API bekerja berdasarkan *events*(kejadian). Kode-kode yang ada pada aplikasi akan memperhatikan suatu *events* pada objek *WebSocket* untuk mengatasi apabila ada data-data yang masuk dan perubahan pada status koneksi. Aplikasi pada *client* tidak perlu melakukan *poll* terhadap *server* untuk memperbarui data. *Events* dan pesan-pesan lainnya akan diterima secara bersamaan saat server mengirimnya.

Sebuah objek *WebSocket* dapat mengirimkan empat *events* yang berbeda, yaitu :

- Open
  Setelah *WebSocket server* memberikan respon pada *connection request*, sebuah *open event* akan berjalan dan koneksi akan dibuat. Setelah hal tersebut terjadi, maka *server* telah berhasil melakukan koneksi dan siap untuk mengirim atau menerima suatu pesan dari aplikasi *client*.

- Message
  Setelah berhasil melakukan koneksi pada *WebSocket server*, maka sudah bisa mengirim atau menerima pesan tertentu. *WebSocket API* akan menyediakan suatu fungsi *onmessage* untuk mengolah pesan yang utuh untuk dikirimkan.

- Error
  Saat terjadi kesalahan yang disebabkan oleh suatu hal, *error event* akan dieksekusi. Saat kesalahan terjadi, dapat diasumsikan bahwa koneksi *WebSocket* akan tertutup dan *close event* akan dieksekusi.

- Close
  Event ini akan dieksekusi pada saat sebuah koneksi *WebSocket* akan ditutup. Setelah koneksi ditutup, maka komunikasi antara *client* serta *server* tidak akan berlanjut.

2.1.5 WebSocket Methods

Suatu objek *WebSocket* memiliki dua *method*, yaitu *send()* dan *close()*.

- send()
  Setelah aplikasi melakukan koneksi menggunakan *WebSocket*, maka aplikasi tersebut dapat mengirimkan metode *send()* selama koneksi tetap terbuka. Sebuah aplikasi akan menggunakan *send()* untuk mengirimkan pesan dari *client* menuju *server*.

- close() Koneksi *WebSocket* dapat dihentikan dengan menggunakan *close() method*. Setelah *method* ini dipanggil, tidak akan ada data yang dikirim atau diterima dari koneksi tersebut. Dalam menutup sebuah koneksi, *close() method* dapat menerima parameter berupa indikasi kepada server yang menunjukan mengapa koneksi tersebut ditutup. Hal itu dilakukan agar dapat lebih mudah dibaca oleh pengguna.

2.2 Socket.io

Socket.io merupakan *JavaScript library* yang digunakan pada aplikasi web untuk melakukan koneksi secara *realtime*. Teknologi ini memiliki dua bagian *library*: bagian *client* yang dijalankan pada *web browser*, dan bagian *server* yang jalankan untuk *Node.js*. *Socket.io* memiliki fitur-fitur yang beragam, seperti melakukan *broadcast* ke beberapa *sockets*, dan menyimpan data yang berhubungan dengan masing-masing *client*.

2.2.1 Connection

Koneksi yang dimulai oleh *Socket.io* dilakukan dengan cara *handshake*. Hal tersebut merupakan bagian yang penting dalam protokol. *Handshake* hanya dilakukan hanya pada saat memulai koneksi, pesan-pesan atau hal lain dalam protokol akan dikirimkan melalui *socket*.

2.2.2 Messages

Apabila koneksi telah dilakukan, seluruh komunikasi antara *client* dan *server* akan menggunakan pesan(*message*) melalui *socket*. Pesan yang akan dikirimkan harus diubah kedalam format yang sudah dispesifikasi oleh *socket.io*.

Format yang sudah dispesifikasi oleh *socket.io* bertujuan untuk menentukan jenis pesan dan data yang dikirimkan dalam pesan tersebut. Format pesan yang sudah dispesifikasi yaitu seperti berikut :

[type] : [id] : [endpoint] (: [data])

- *type*, merupakan satu digit angka integer yang menunjukan jenis pesan yang akan dikirim.

- *id*, merupakan identitas pesan yang terdiri dari beberapa digit angka.
- *endpoint*, merupakan *socket* tujuan yang akan menerima pesan yang sedang dikirim. Apabila tidak ada *endpoint*, maka pesan akan dikirimkan ke *default socket*.
- *data*, merupakan data yang akan dikirim ke *socket* tertentu. Pada kasus *messages*, data akan dikirimkan dalam bentuk *plain text*, sementara pada kasus *events* akan dikirimkan dalam bentuk *JSON*.

2.3 Node.js

Node.js merupakan sebuah platform yang didesain untuk mengembangkan aplikasi berbasis web pada bagian *web server*. *Node.js* ditulis dalam sintaks bahasa pemrograman *JavaScript* dan memiliki sifat *non-blocking* yang berarti *Node.js* tidak akan menunggu untuk mengerjakan *request* selanjutnya. *Node.js* dapat berjalan pada berbagai sistem operasi, seperti *OS X*, *Windows*, dan *Linux*. Dengan begitu, tidak ada batasan dan perbedaan dalam menjalankan fungsi-fungsi yang ada pada berbagai sistem operasi.

2.4 Express.js

Express.js merupakan sebuah *framework* aplikasi web untuk *Node.js*. Express.js menyediakan fitur-fitur yang membuat pengembangan aplikasi web dapat bertahan lama. Teknologi ini pun merupakan modul *node package manager (npm)* yang menjadi ketergantungan dalam suatu aplikasi. Agar dapat berjalan, seluruh *file* yang dimiliki oleh *framework* ini harus berada pada *node_modules* lokal dalam suatu projek tertentu.

2.5 Canvas API

Canvas API merupakan salah satu elemen *HTML5* yang digunakan untuk membuat gambar grafis dalam aplikasi web. Teknologi ini memiliki fitur untuk membuat komposisi foto, membuat animasi, dan membuat *real-time video processiong* atau *rendering*. Untuk dapat menggunakan elemen *canvas* harus menambahkan *tag* <canvas> pada suatu halaman *HTML*. *Tag* <canvas> memiliki tiga atribut utama dimana atribut tersebut terdapat didalam kurung lancip pada *HTML tag*. Atribut-atribut tersebut yaitu :

- id
  Merupakan nama yang akan digunakan sebagai referensi dalam kode JavaScript. Dimana nantinya nama tersebut akan merujuk ke *tag* <canvas> yang memiliki nama yang sama.
- width
  Merupakan lebar dari *canvas* yang dibuat.
- height
  Merupakan tinggi dari *canvas* yang dibuat.

Menggunakan *Canvas API* membutuhkan dasar yang kuat dalam menggambar , dan merubah bentuk-bentuk dasar dua dimensi. Berikut merupakan bentuk-bentuk dasar dua dimensi yang dapat digambar pada *canvas*.

- Rectangle
  Untuk menggambar suatu *rectangle* (persegi), *canvas* menyediakan tiga *method* yaitu sebagai berikut:

  - fillRect(x,y,width,height)
    Menggambar persegi dengan warna yang penuh mengisi bagian dalam persegi pada posisi x,y dengan ukuran persegi *width* dan *height*.
  - strokeRect(x,y,width,height)
    Menggambar garis luar persegi pada posisi x,y dengan ukuran persegi *width* dan *height*.
  - clearRect(x,y,width,height)
    Mengosongkan area tertentu dan membuat area tersebut transparan pada posisi x,y dengan ukuran persegi *width* dan *height*.

- Paths
  Paths merupakan *method* yang digunakan untuk menggambar seluruh bentuk pada *canvas*. Path merupakan kumpulan titik, dan garis yang digambar diantara titik-titik tersebut. Untuk menggunakan path pada *canvas*, dibutuhkan dua fungsi utama. Fungsi tersebut yaitu beginPath(), yang akan mulai membuka suatu path pada canvas, fungsi lainnya yaitu closePath(), yang akan menutup suatu path pada canvas.

- Arcs
  Sebuah *arc* (garis lengkung) dapat berupa suatu lingkaran utuh atau bagian dari lingkaran tertentu. Untuk menggambar sebuah garis lengkung, *Canvas API* menyediakan beberapa fungsi yang dapat digunakan. Salah satu fungsi tersebut yaitu:

  arc(x, y, radius, startAngle, endAngle, anticlockwise).

  Nilai x dan y merupakan titik pusat dari lingkaran, dan radius merupakan jarak dari titik pusat ke suatu titik tertentu dimana garis lengkung akan digambar.*startAngle* dan *endAngle* ada dalam satuan radian, bukan derajat. *anticlockwise* merupakan suatu *boolean* yang menandakan apakah garis lengkung tersebut akan searah jarum jam atau tidak.

Selain menggambar suatu bentuk tertentu, pada *canvas* pun dapat memberi warna pada bentuk yang sudah dibuat. *Canvas API* memiliki properti yang digunakan untuk memberi warna dasar pada bagian dalam suatu bentuk di *canvas* yang bernama fillStyle. Contoh penggunaan properti tersebut sebagai berikut:

context.fillStyle = ”red”;

Langkah tersebut akan memberikan warna merah pada suatu bentuk tertentu. Selain itu, ada beberapa cara yang dapat dilakukan untuk memberikan warna dasar pada suatu bentuk. Cara tersebut dijelaskan sebagai berikut:

context.fillStyle = ”rgb(255,0,0)”;

Method rgb() akan menggunakan nilai RGB 24-bit pada saat memberikan warna pada suatu bentuk tertentu.

context.fillStyle = ”#ff0000”;

Properti ini dapat menerima bilangan hex dalam bentuk *string*.

context.fillStyle = ”rgba(255,0,0,1)”;

Method rgba() akan menggunakan nilai 32-bit dengan nilai 8 bit di akhir yang merepresentasikan nilai *alpha* pada suatu warna.

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```c

// This does not make algorithmic sense,
// but it shows off significant programming characters.

#include<stdio.h>

void myFunction( int input, float* output ) {
    switch ( array[i] ) {
        case 1: // This is silly code
            if ( a >= 0 || b <= 3 && c != x )
                *output += 0.005 + 20050;
            char = 'g';
            b = 2^n + ~right_size - leftSize * MAX_SIZE;
            c = (--aaa + &daa) / (bbb++ - ccc % 2 );
            strcpy(a,"hello $@?");
    }
    count = ~mask | 0x00FF00AA;
}

// Fonts for Displaying Program Code in LATEX
// Adrian P. Robson, nepsweb.co.uk
// 8 October 2012
// http://nepsweb.co.uk/docs/progfonts.pdf
```

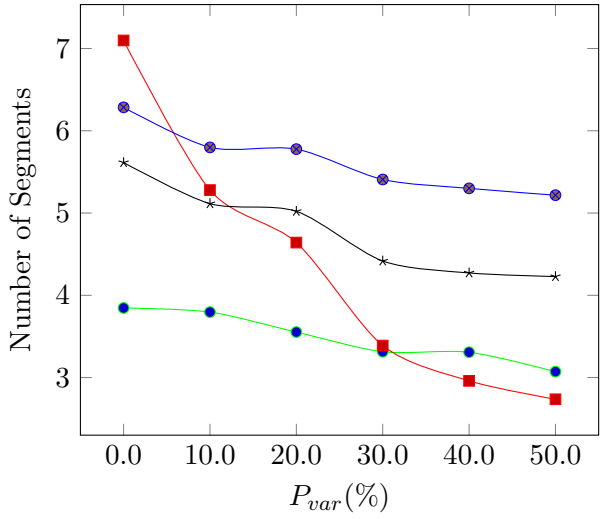
Listing A.2: MyCode.java

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                   //id of the set
    protected MyEdge FurthestEdge;                      //the furthest edge
    protected HashSet<MyVertex> set;                    //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;    //list of all vertices in the set for each trajectory
    protected ArrayList<Integer> closeID;               //store the ID of all vertices
    protected ArrayList<Double> closeDist;              //store the distance of all vertices
    protected int totaltrj;                             //total trajectories in the set

    /*
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

}
```

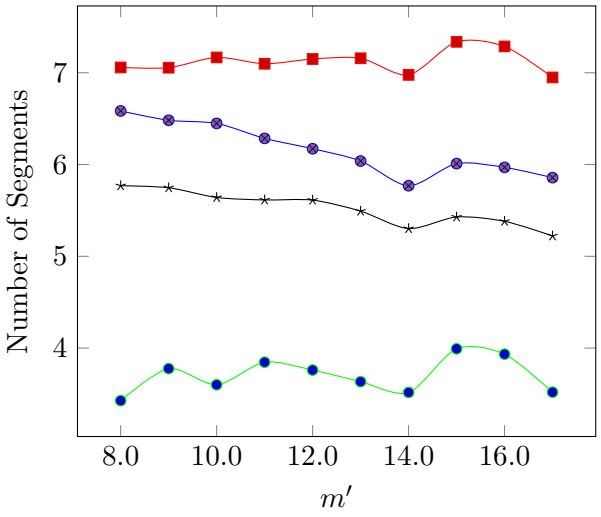
LAMPIRAN B

HASIL EKSPERIMEN

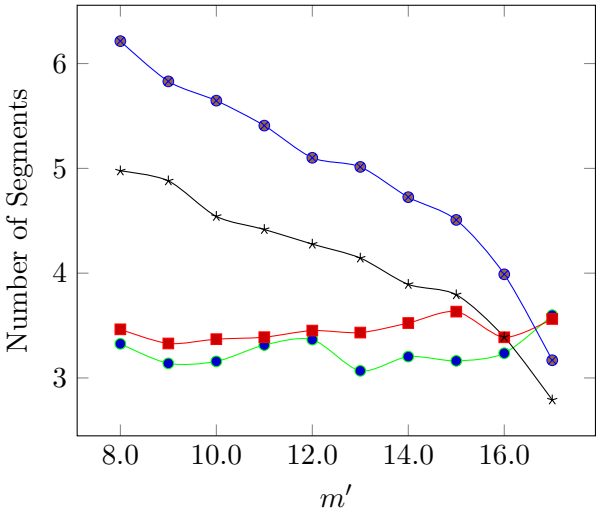
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



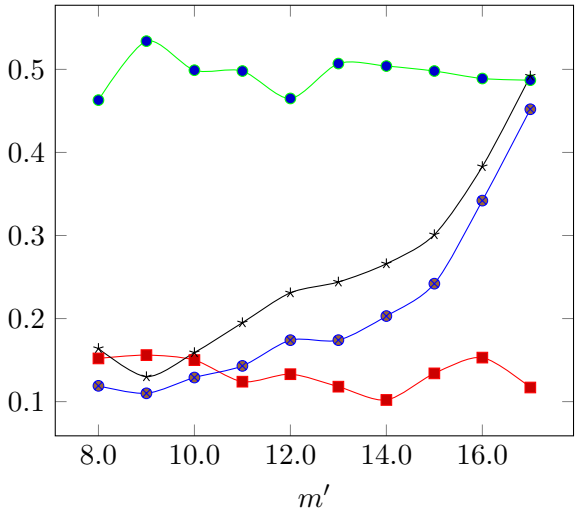
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4