

Bab I. Pendahuluan

“*Omnia causa fiunt, segala sesuatu pasti mempunyai penyebab.*”
- pepatah Latin –

1.1 Latar Belakang

Sebuah sistem dinyatakan reaktif apabila mempunyai sifat mempertahankan interaksi ke lingkungan dan menanggapi stimulus secara kontinu [3]. Sebuah sistem reaktif menghasilkan nilai melalui proses interaksi yang berkelanjutan, bukan melalui proses *termination*. Beberapa contoh sistem reaktif adalah *elevator system*, *air-traffic control system*, dan program yang mengendalikan proses berkesinambungan seperti program pengontrol reaktor nuklir. Beberapa jenis game, terutama yang mempunyai *genre arcade*, termasuk salah satu contoh sistem reaktif [2].

Reactive programming merupakan salah satu paradigma pemrograman yang bertujuan untuk menyederhanakan pengembangan sistem tersebut. Dengan paradigma *reactive programming*, suatu program yang bersifat reaktif dapat dikodekan dengan dukungan abstraksi tambahan untuk mendefinisikan *behavior*. *Reactive programming* bersifat *event-driven*, atau pada pengaplikasiannya dikendalikan oleh pengaruh *event* (kejadian), misalnya penekanan tombol atau perubahan sinyal melalui sensor.

Functional programming merupakan paradigma pemrograman tertua yang berbasiskan pada *lambda calculus*. Paradigma ini muncul jauh sebelum paradigma pemrograman imperatif atau *object-oriented*. *Functional programming* memandang program sebagai suatu rangkaian evaluasi fungsi matematika dan biasanya digunakan untuk merancang sistem yang memerlukan kinerja komputasi tinggi, misalnya sistem peluncuran roket ataupun robotika. Apabila sebelumnya hanya digunakan sebatas keperluan akademis, kini dunia industri mulai menggunakan *functional programming* sebagai basis pengembangan sistemnya, misalnya oleh Twitter dan Google.

Functional Reactive Programming (FRP) merupakan paradigma pemrograman untuk sistem reaktif yang memanfaatkan pola pikir *functional programming* sebagai *building blocks*. Paradigma ini memandang sebuah sistem berdasarkan hubungan sebab-akibat. FRP bukanlah merupakan aturan baku, melainkan suatu sudut pandang turunan *functional programming* yang ditujukan secara spesifik untuk domain sistem reaktif. Paradigma FRP telah menjadi salah satu topik *research* yang paling intensif di komunitas *programmer* Haskell sejak tahun 1997 [3].

Selama ini game lazim dirancang dengan menggunakan paradigma *imperative programming*, di mana sebuah program dipandang sebagai rangkaian instruksi yang sekuensial. Tugas akhir ini mencoba merancang beberapa game sederhana dari sudut pandang yang berbeda, yaitu paradigma FRP dan kelayakan dari pendekatan ini akan dianalisis. Hasil tugas akhir berupa beberapa game yang dikembangkan di atas sudut pandang paradigma FRP dan sebuah *library* yang berfungsi mendukung manipulasi *first-class event*. Pada tugas akhir ini juga dilakukan analisis komparasi implementasi paradigma berdasarkan *code size* dan pengujian performa sistem yang dibangun.

1.2 Perumusan Masalah

Perumusan masalah pada tugas akhir ini adalah :

1. Bagaimana merancang game dengan sudut pandang paradigma FRP.
2. Bagaimana menjelaskan sudut pandang paradigma FRP berdasar studi kasus berupa sejumlah game yang telah diimplementasikan.
3. Bagaimana memposisikan (analisis perbandingan) FRP sebagai solusi perancangan game secara relatif terhadap paradigma *imperative* yang lebih tradisional berdasarkan dengan *code size*.
4. Bagaimana korelasi antara jumlah *event* dan performa sistem FRP.

Dari perumusan masalah di atas, hipotesa awal yang diajukan adalah :

1. FRP mampu digunakan sebagai paradigma dalam perancangan game yang bersifat reaktif.
2. FRP merupakan solusi yang lebih *concise* dengan *code size* yang lebih rendah di dalam menangani fungsionalitas-fungsionalitas sistem yang berkaitan dengan *event*, dibandingkan solusi *imperative*.
3. FRP merupakan solusi yang implementasinya mempunyai performansi yang menurun seiring dengan banyaknya *event* yang didekalarasikan.

1.3 Batasan Masalah

Sedangkan batasan masalah yang digunakan di dalam tugas akhir ini adalah :

1. Teknologi yang digunakan di dalam keseluruhan tugas akhir ini adalah Microsoft .NET Framework. Game berbasis paradigma FRP dirancang di atas bahasa pemrograman F# yang berbasis multi-paradigma.

2. Game yang dirancang merupakan tiga macam game sederhana berbasis pada game populer yaitu Space Invasion, Bouncing Balls, dan Racecar.
3. Game Space Invasion merupakan implementasi primer di dalam tugas akhir ini, sedangkan dua game lainnya berperan sebagai komplemen.
4. Perancangan game difokuskan secara intensif pada fungsionalitas yang berhubungan dengan penanganan *event*. Hal-hal komplementer seperti desain *user-interface*, aplikasi *artificial intelligence*, manajemen basis data, strategi algoritmik, dan lain-lain tidak menjadi fokus penelitian.
5. Data analisis dan kesimpulan yang diambil terbatas kepada game-game yang diimplementasikan.
6. Analisis *software metrics* yang diaplikasikan terbatas pada *conciseness* secara *physical* maupun *logical*.
7. Kode yang dibandingkan di dalam analisa *code size* berlandaskan pada fungsionalitas-fungsionalitas primitif pada .NET 4.0.
8. Penilaian game yang dirancang tak dilihat dari sisi *playability*, *beauty*, kompleksitas cerita, atau tingkat kesulitan.
9. Tugas akhir berfokus pada paradigma atau sudut pandang, bukan pada sisi teknis seperti cara memakai *tools*, cara meng-*install*, cara membuat *project*, cara meng-*compile*, dan cara men-*deploy* aplikasi.

1.4 Tujuan Penelitian

Adapun tujuan dari penelitian dan penulisan laporan tugas akhir ini adalah :

1. Memberikan penjelasan garis besar sudut pandang paradigma FRP dan panduan implementasinya di dalam merancang game.
2. Membuktikan bahwa paradigma FRP sanggup menjadi sudut pandang perancangan game dengan mengimplementasikan paradigma tersebut pada penanganan fungsionalitas game, terutama untuk *event-handling*.
3. Melakukan pengukuran terhadap implementasi paradigma FRP secara relatif terhadap paradigma *imperative* dari sisi *conciseness*.
4. Melakukan *testing* untuk menemukan relasi antara jumlah *event* yang dideklarasikan dan performa sistem berbasis FRP.

1.5 Metodologi Penelitian

1. Identifikasi Masalah

Mengumpulkan literatur dan *source code* tentang metode perancangan game yang tradisional dan paradigma lainnya.

2. Studi Literatur

Mempelajari perkembangan terkini teknologi *functional programming* dengan menggunakan F# yang pada saat proposal ditulis masih berada dalam tahap pengembangan.

3. Perancangan Prototipe

Melakukan eksperimen dengan mengkodekan berbagai prototipe game dengan menerapkan konsep-konsep paradigma FRP.

4. Perancangan Model

Merancang model matematis dan mencari diagram-diagram yang dapat mendukung pemodelan sistem berbasis FRP.

5. Perancangan Game

Merancang game dengan berdasarkan kepada model yang telah dibuat dan prototipe yang dihasilkan sebelumnya.

6. Pengujian dan Analisa

Melakukan pengujian (*benchmarking*) dan analisa terhadap game yang telah dirancang sebelumnya.

7. Penyusunan Laporan

Menyusun laporan tertulis tentang hasil penelitian serta menyimpulkan hasil penelitian tersebut dan memberi saran pengembangan.

1.6 Sistematika Penulisan

BAB I PENDAHULUAN

Berisi pemaparan mengenai latar belakang masalah, tujuan, perumusan masalah, batasan masalah, metodologi, dan sistematika penulisan.

BAB II LANDASAN TEORI

Berisi uraian tentang landasan teori dan contoh-contoh yang memberi penjelasan terhadap konsep paradigma FRP, meliputi *lambda calculus*, *functional programming*, dan *functional reactive programming*.

BAB III PERANCANGAN SISTEM

Berisi mengenai perancangan suatu game sebagai contoh implementasi paradigma FRP.

BAB IV PENGUJIAN DAN ANALISA

Berisi mengenai pengujian terhadap *conciseness* serta performa sistem yang dirancang. Ditambah dengan analisa terhadap hasil pengujian.

BAB V PENUTUP

Berisi kesimpulan dan saran untuk penelitian yang lebih lanjut.