January 23, 2017

PROCESS REPORT

Creating a pipeline application

Fontys – EI3T1 & EI3S1

Fabian van Amerongen
Maximilian Bogdanov
Veselin Slavchev
Nguyen Le

January 23, 2017

Table of Contents

Introduction **3**

Understandings **3**

Division of labor **3**

Program choices **4**

Encountered problems **5**

Evaluation **6**

Introduction

This report will discuss the progression of the pipeline application and how it got to its final state. Also, evaluation, recommendations and personal views will be discussed.

Understandings

The teacher and group have agreed on the group making an application that allows the user to build a simulation of a pipeline network. This application will be written in Java and will be able to run on all OS's that run JRE 8. The application is meant for non-mobile devices like laptops and desktops. The final application will be presented on Wednesday the 25th of January at 11:00.

Division of labor

In working on the application, four main deliverables have been set.

1. User Requirements Specification (URS)
2. Design Document
3. Source Code
4. Process Report

All four have been accomplished as a group effort. With some deliverables, some group members have been contributing more than others though.

1. **User Requirements Specification (URS)**
   The functional requirements have been written by Veselin, Use-cases were mainly done by Max and Fabian with some help from Nguyen. The non-functional requirements were written by Nguyen and the user interface was made my Fabian. All was discussed in group meetings and approved by all group members. Fabian combined all information and bundled it into one document.

2. **Design Document**
   In making the design document, Veselin focused mostly on the class diagram which he finished with Max. Nguyen and Fabian worked on making the sequence diagrams. Again the class diagrams and sequence diagrams have been discussed thoroughly in several meetings and are approved by all group members. Veselin and Nguyen created the final document.

3. **Source Code**
   The source code is written by all of us. Veselin has been the biggest contribution to this, followed by Nguyen, Max and Fabian (decreasing order).

   Most of the initiation on the project, insight on user experience/interaction with the program, and the documentation was done by Fabian. Therefore, his contribution to the code was more limited, however he dealt with the graphics aspect of our solution: rendering of the objects, setting up the UI with buttons and bars and configuring mouse clicks and cursors, and tackling with the colorful display of the pipes, which is undoubtedly the prettiest part of our application.

   Maximilian proved very helpful by working on the serialization, which was more complicated than expected, and finding creative ways to crash the application and afterwards coming up with simple rules to prevent that. Everybody can come up with a complex solution to a complex problem, but the simpler ones are always more robust and stable, and the way Max solved those impacted positively the complexity of our codebase.

   Nguyen was the most skilled at math from all of us, so he had a lot of fun dealing with the more algorithmic problems: implementing our own version of a tree data structure for our needs and all the methods we needed to use if properly, complex rules with traversing the tree, and collision checking for the pipes. Last one was giving us bugs that broke important parts and we sadly decided to not include it in the final version, however we will finish it for the challenge after the subject is done.

   Veselin did most of the architectural design and planning of the overall structure of the solution. He came up with a few ways to connect all our desired functionality and divide the responsibility of the solution into the classes, using careful preparation; he also enforced clean code and refactoring of some ugly code to keep it readable, understandable and maintainable. This made fixing bugs easier and implementing new functionality faster and less buggy.

4. **Process Report**
   The process report was mainly written by Fabian, of course with input from Veselin, Nguyen and Max.

Program choices

During programming, we encountered very few situations that were not covered by our previously written documentation. One situation we did encounter though was the direction of flow in a pipe. The application is now written in a way that the flow is always in the direction of the way the pipe is drawn. Preferably we would have it otherwise. We addressed the issue, discussed it during a group meeting, but unfortunately had no time to implement our found solution. Due to this, a functional requirement becomes that users must draw pipes in the direction of the flow. Besides this, most other code is written in compliance with our documentation.

Encountered problems

While doing this project, some problems were encountered. An issue we had at the beginning was focusing on documentation other than coding. For a first time, we were forced to first think out the entire application and wait with programming until everything was documented. This is different from previous projects we worked on so this was quite a challenge for us. Eventually this resulted in a very well documented application that was fairly easy to program due to not having to think about certain functionality anymore.

The organization of the classes had quite a few changes when making the design document. We were not satisfied with the first 3 or 4 designs so we rearranged classes, abstractions, contracts and sharing of responsibilities of the methods. After much revisiting, we came up with a nice balance between code-reuse, flexibility, ease of understanding and coupling of classes.

The programming phase was hard on us due to its timing. The design document had to be delivered a few days before the Christmas break and some of us have been out of town during the Christmas holiday. This made communication and timing difficult. This also considering the upcoming exams directly after the holidays. Timing and communication have been a learning point and I think we all now have a better idea on how to approach a similar situation next time.

During coding, we also encountered some issues: Serializing the simulation class was problematic because the components had reference to images. The references themselves were not static, but the objects in memory were static, as they were loaded by the image manager, which was a static class. So, we had to change the way a component knows which image it must fix. But the reason for the bug was hidden and took us some time to fix.

The algorithm for checking collisions with the pipes was working perfectly for simple pipes (1 segment), but when the pipe consisted of multiple lines we could not adjust it to check all of them correctly.

Another challenge came from the initial use of swing for the UI, as some features of the framework were not working properly. We then found out that swing is no longer supported by Oracle and the new tool to use was JavaFX. So, we started learning it and redid all our progress with it. The result was very satisfying as javaFX was much more beautiful, robust and easy to set up. The code also took less lines.

Evaluation

Max
I consider this project to be really beneficial for my development. I'm happy with the fact that we picked an unfamiliar for all the group members language (Java) and still managed to get things to work the way we wanted. This taught me that having a specific goal is enough for you to create something from scratch even using new technologies. I believe that the project improved my qualities as a team member and initiator. An issue that I had was with object serialization. Turns out images are non-serializable. I think the assignment was great and definately challenging.

Nguyen
I had learned a new language, which is Java, and create a program with my groupmates. One of the problem we have is leaving things until the deadline, which results in lots of rushed decisions. One of those decisions that I regret most is deleting the optional part about collision objects (which works at the beginning) because it somehow affect other new codes and make the program stop working properly. We decide to eliminate it because of the deadlines. Overall, in my opinion, this is a good assignment which improving our teamwork, planning-designing-building phase with various programming skills (ex. graphics, performance, complex rules)

Veselin
We chose to use color coded pipes to show the flow/capacity ratio in an intuitive way. We used Java 8 because we accomplished more with less lines of code; that means less time spent staring at code and easily understandable behaviour. We used javaFX because it looked better and hade better support for some of our needs such as file dialogs, canvas resizing and clicks on UI buttons. I learned to utilize another high-level language and build a system with lots of complex rules with my teammates. And, that we shouldn't leave things for the last moment and be more careful with deadlines. I think this is a very good assignment for students as it challenges us with teamwork, planning, design decisions, and various programming topics like graphics, performance, complex rules; and of course, makes us better people by requiring from us to be patient, humble, leading, active, curious and most importantly ever improving.

Fabian
The biggest learning point from this project was understanding the value of good documentation to program more effectively. Class diagrams and sequence diagrams are a good fallback when coding and make it more clear to work on.
The choice of working with Java over a programming language we are already familiar with gave me the confidence that with a good understanding how a programming language works, you can program with most programming languages. The basics are usually the same and work for all languages.
As of a team project it was good to work with dedicated people. I liked how we handled certain issues and gave each other constructive feedback during meetings.