Final Report [Draft]

Programming Life
Team Dynamites & Butterflies

Jasper van Tilburg - 4394554
Marc Visser - 4318234
Lex Boleij - 4153197
Eric Dammeijer - 4412265
Jip Rietveld - 4486528

22-6-2017
TU Delft

Table Of Contents

Introduction, including a brief problem description and end-user's requirements.

In the following report we will describe the results of our Context Project “Programming Life”. Our group, “Dynamite & Butterflies”, consists of five people, who together spent a little over 2 months developing an application that can visualize multiple genomes as a graph. The main problem presented was that the tools that currently exist for this visualization are too slow, which makes those applications not very useable/interactive.

Our application thus has to be fast and dynamically, with respect to the following (most important) requirements:

- Loading of big files (up to millions of nodes).
- Dynamic zooming and panning (fast).
- Being able to use what is on screen (the context of the graph/click on things).
- Make sure the graph looks nice with minimal edge crossings.

These three requirements are the things we focussed on most, as these are essential for a working and useable program. Of course we implemented lots of other features, which will be discussed later in the report.

Overview of the developed and implemented software product

There are a lot of tools on the market that can visualise genomes. However these tools all have their flaws. Most tools can only handle single genomes, and the tools that can handle multiple genomes are often slow. This created the opportunity for us to develop a tool that can handle multiple-genomes, and is fast enough to interactively pan across entire genomes.

The program works by loading Graphical Fragment Assembly (GFA). The program will convert the GFA file to a graph structure. The program uses a Layered graph drawing algorithm to create a visual representation of multiple genomes. When the data is loaded the user can navigate the data in multiple ways. They can use the the arrow keys or panning buttons to dynamically pan across the genomes, a go to node button is provided to go to a specific location and the user can click on the minimap. Visual encodings can be enabled this allows the user to easily follow which genomes go through which node

Furthermore, features were developed such that the user can go to locations relevant for their research. The user can load general feature format (GFF) files. These annotations allow the user to more clearly see what part of genomes make up specific traits. The user can then save this location with a bookmark option, for later analysis.

Reflection on the product and process from a software engineering perspective

Development Process (Scrum)

For the duration of this project, we used the Scrum method as our development process. By using this, we were able to manage our time and tasks efficiently during each one-week sprint. Mainly in the first few weeks we had some trouble estimating the required time for certain tasks, as this project involves a lot of problems that look relatively easy, but in the end, are pretty hard to solve as it has to be (one of) the fastest solutions available to keep the program dynamically and fast. But when we got used to estimating some tasks this (exceptions aside) went a lot better as we learned from earlier sprints. Usually the division of tasks was pretty straightforward and all tasks were reasonably even divided, where everyone chose the tasks where his strengths lie.

Tooling

During the development of our program we used multiple tools to help us aid our development process. Our program is structured as a maven project, which enforces a consistent file structure, build process and helps with the integration of various analysis and build tools. The tools we used are github, travis-ci, checkstyle, pmd, findbugs, and sig. For version control and continuous-integration we coupled our github with travis. Travis will build every new push to the version control system in a clean environment, informing us about failing tests, build errors, and missing dependencies. We used intellij's checkstyle plugin to allow for a custom checkstyle to enforce a consistent coding style. Furthermore we used PMD and FindBugs to help us find bugs due to bad programming patterns or unused variables which may go unnoticed. Another tool we used is SIG to check our code for bad practices. The tool analyses the code and looks at 10 metrics to determine the quality of our code, and allowed us to adjust our code accordingly.

Software Design

In this section we will discuss the way we designed our software product.
We made extensive use of software patterns in order to make our code clean, reusable and extensible. We implemented a model view controller pattern, the model consists of the parser and the sequence graph, the controllers handle how the UI works based on the model, and the view updates the model (figure 1). They communicate with each other using a observer pattern. Furthermore the Drawable canvas and graph drawer are singletons, as we only want a single instance of those objects. We haven’t made any interfaces, this is due to the nature of our program. It is basically a graph with nodes, which means not many different kinds of objects are used. This makes programming to an interface less useful.

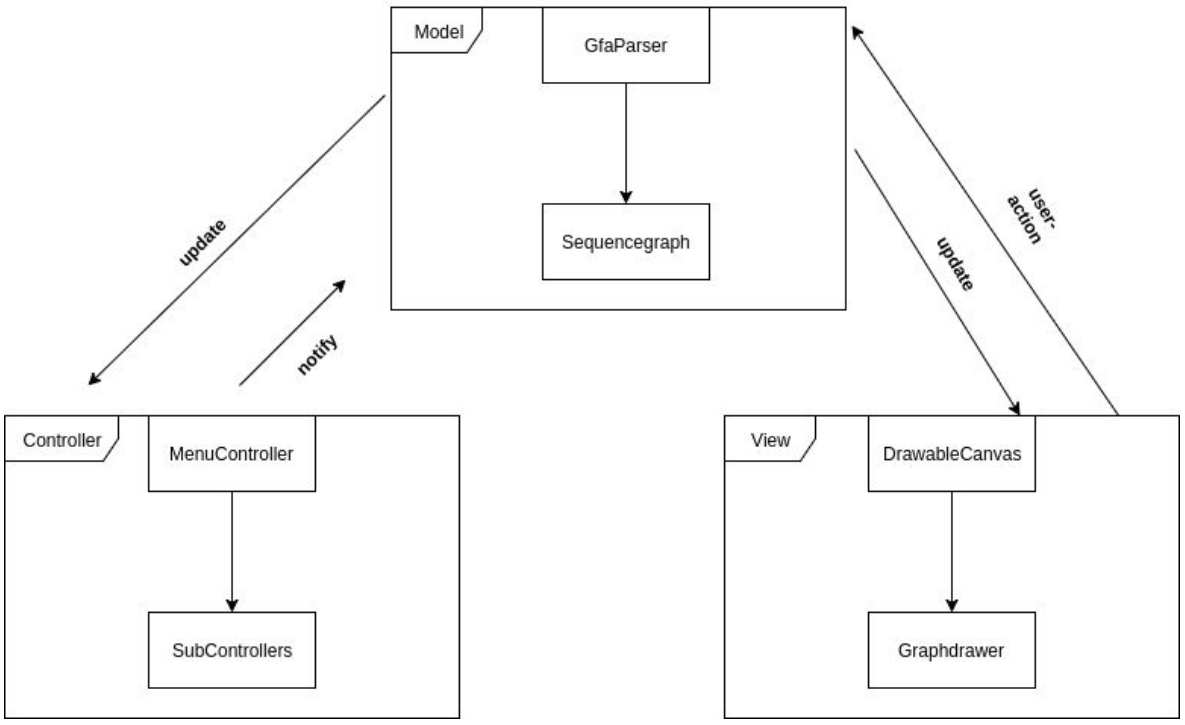

Figure 1. MVC, Observers

Description of the developed functionalities

In this section we will give a brief overview of the developed software product. The software we developed has three major components, the loading of big data sets, the visualisation of the data, and the navigability and usability of the application.

Data loading / parsing

One of the biggest tasks when making this software product was being able to load and display big gfa files efficiently. The biggest file we had to parse was 2.9GB of text. This file contains data for how the graph should be structured, the dna sequences in each node, and which genomes go through which sequences.

Loading the data structure

In order to load the data structure efficiently and quick memory proved to be an issue. We first tried a mapDB implementation which stores the data in a hashmap like structure in a persistent database which can be queried. The problem however was that writing this data to a db file was too slow, we ended up with multiple files as large as 8gb which took 15 minutes to write. The solution we came up with was saving the edges of the graph in memory, this reduced the parsing time dramatically, from there we created “sub-graphs” from specific edge intervals.

Loading the genomes / sequences

When loading the gfa file, the biggest part of that file are the sequences. As we are unable to hold this in memory, for this we use MapDB to create a database on the user's computer, which we can access quite fast. As we only make a call to that database when we need to show the information of a specific node, this works quite well.

Loading the annotations

For extra information about the genomes, we are able to open gff files. When opened, the user is asked which genome this information is about (we do give the suggestion for a genome we think it should be linked to by looking at a specific column in the file). When these annotations are loaded, the user can search/filter through all the annotations and jump to a specific annotation.

Data integrity

Another issue we encountered was data integrity. When a user closes the program during the parsing of the data the mapDB files get corrupted. Next time this user starts the program a notification will pop up telling the user that the data needs to be reloaded as the file is corrupt.

Visualisation

The second major aspect of our software product is visualising the data, such that the user can easily look at, and analyse the dataset. We achieved this goal by implementing a graph structure with minimal edge crossing, dynamic zooming and panning, edge and node thickness depending on genomes, color coding genomes, annotations and a compare tool.

Displaying the graph structure

In order to properly display the data clearly multiple additions had to be made to the data structure. According to Sugiyama, K. (2002), the following steps are required to uniformly distribute nodes en minimize edge crossings in a graph:

“Step I: Making the general directed graph acyclic.
Step II: The assignment of vertices to layers in the acyclic directed graph.
Step III: The determination of the order of vertices on each layer.
Step IV: The determination of the position of vertices on each layer.” (p. 61)

For Step II we will use the procedure described by Tamassia, R. (2010), where he describes how to layerize graphs using the longest-path algorithm and the use of dummy nodes in order to reduce edge crossings.

For Step III we will use one of two methods described by Matuszewski C., Schönfeld R. and Molitor P. (1999) . “The most popular Heuristics for one sided crossing minimization are the barycenter and the median heuristic. There are other heuristics known from literature, …, are mostly outperformed by barycenter and median heuristics.” (p. 218). The idea of barycenter and median heuristics is further explained by Patarasuk, P (2004): “The barycenter heuristic only needs to calculate barycenter values for each vertex, and then sorts the vertices

according to these values. Hence, no comparison between numbers of crossings is made." (p. 18).

After implementing all the steps we had a graph as seen in figure 2.

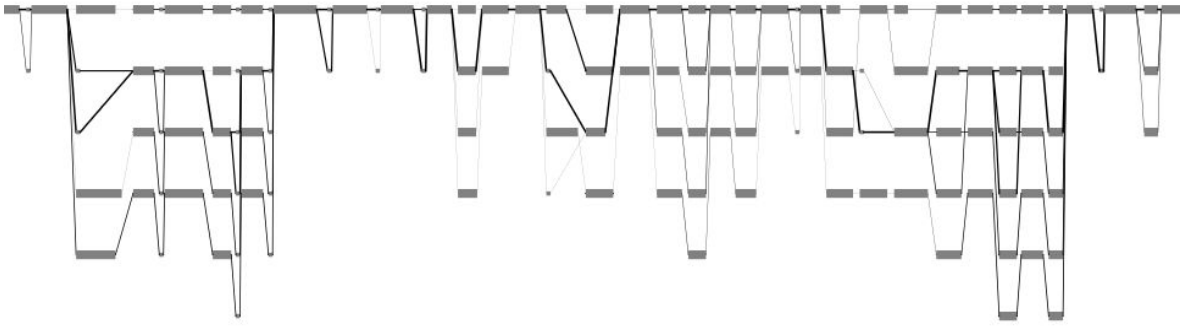

Figure 2. Graph visualisation

The edge crossings are reduced to a minimum and the path of each sequence can be followed easily, creating a graph which can be easily analysed.

Node information

Edge and node thickness

Another feature we added to better analyse the data is edge thickness and node length as seen in figure 1. The length of the nodes depends on the length of the sequence, because sequences can differ in length a lot and we don't want a single node taking in the whole screen we opted for a logarithmic scale. The same approach was taken for genomes passing through edges. By doing this the end-user can see which nodes/edges have the most genomes passing through them and can estimate the length of sequences in the nodes.

Displaying color coded genomes

Color coding was implemented for genomes. This was done so that the end-user can easily track which nodes are in which specific genome. We added an option for rainbow mode (figure 3), which is useful if there are only a couple of genomes we want to look at. The other option is opacity mode (figure 4), which is doesn't show specific genome but changes the opacity based on the amount genomes going through them.

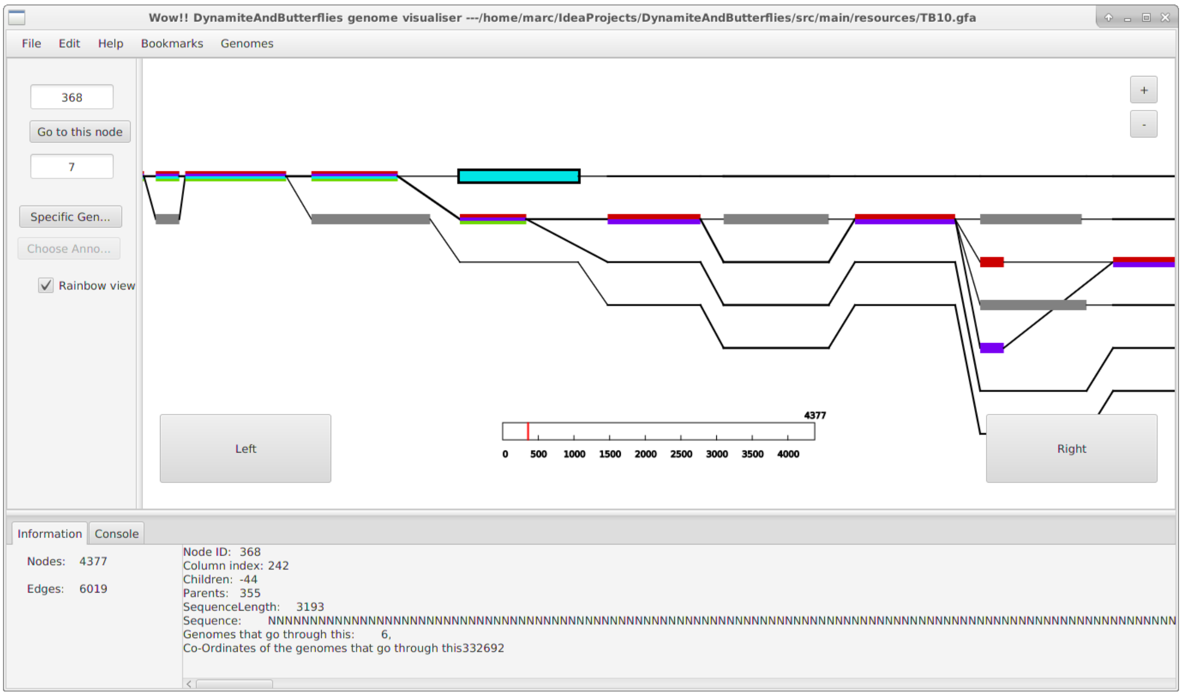

Figure 3. Rainbow mode.

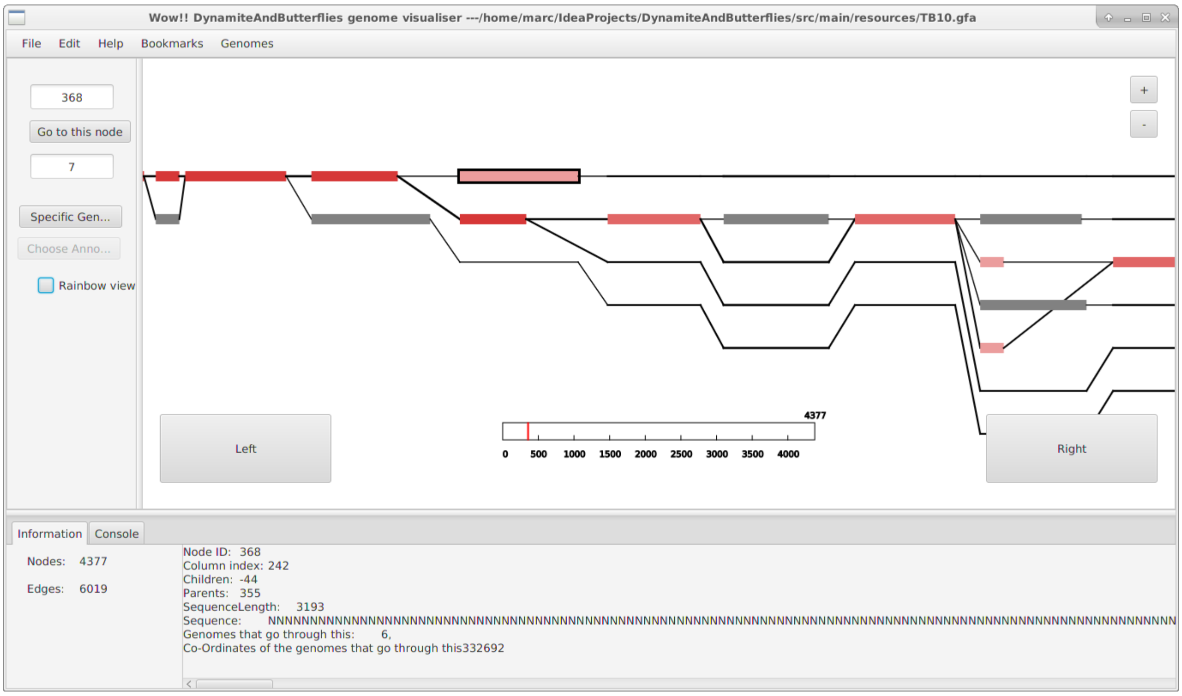

Figure 4. Opacity mode.

Displaying annotations

The annotations are all displayed when loaded and linked to a specific genome. This is done using a coordinate system corresponding to the number of the base pair in that specific genome. This system is pretty slow to instantiate, as the coordinate system of the graph

itself is different, but once it is loaded, the lookup is fast. The annotations are displayed as red lines beneath the node as seen in the figure underneath.
TODO: Add Figure
TODO: Let the annotations follow the edges too.

Semantic zooming

In the data there are a lot of “snip” structures, waiting for final implementations.

Navigation / Usability

Go to node

The first navigational feature we implemented was the go to node function, in the left panel as can be seen in figure 2 and 3, there are two text boxes. The first textbox corresponds to a node identification number, and the second box refers to a range. One can simply enter these values and press on the button and the graph will shift to the specific node ID. This feature is more of a stepping stone for other features, this is because the end-users have no way of knowing which node ID is which sequence beforehand. This is why bookmarks were added.

Bookmarks

When the end-user sees an interesting genome/annotation/structure he might want to go back and look at it later. The user can simply press on the bookmark button and a window will pop up. He will be asked for a description, and he can change the node ID and range of which he wants a bookmark. The user can now select and go to this specific bookmark from the bookmark option in the menubar.

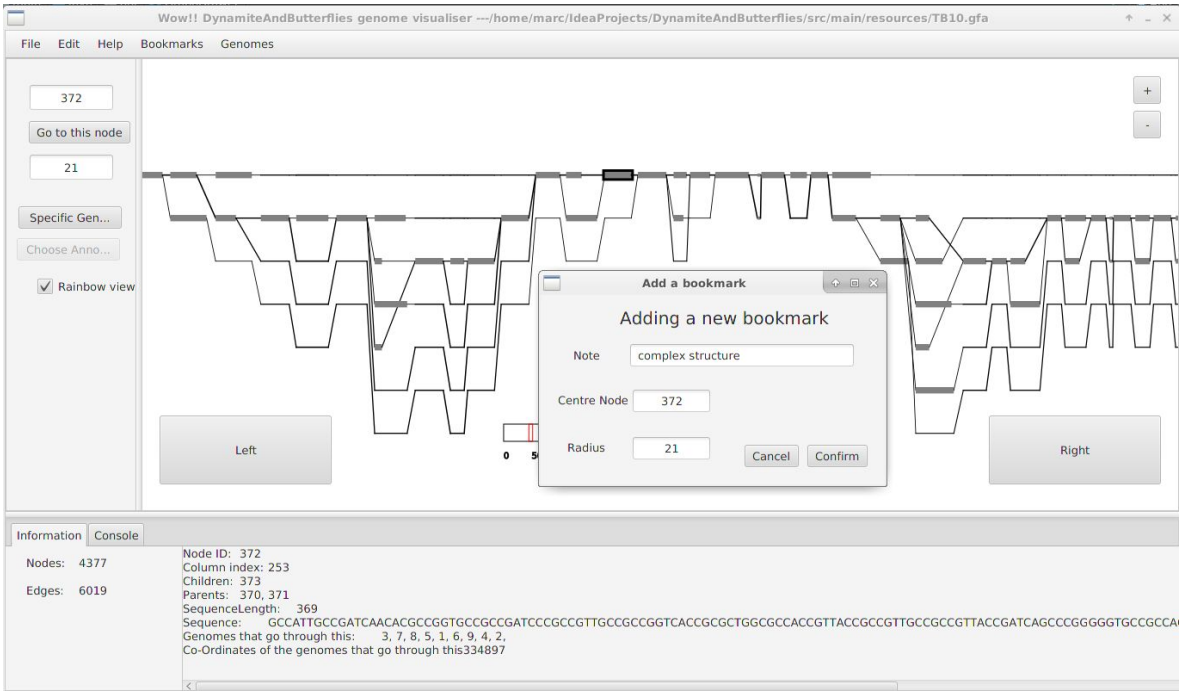

Figure 5. Bookmarks

Dynamic zooming and panning

In order for the end-user to explore the data he will need to be able to effortlessly pan across the data and be able to zoom in and out without stuttering. We used the subgraph functionality we had previously implemented and added a threshold based on the nodes on screen. For example if we are at node 8000 with a range of 3000, it will start to make a subgraph of node 5000 to node 11000. If the user now pans across these nodes and hits a certain threshold node on screen, a new subgraph will start to load in the background and the display graph is updated.

Minimap

In order for the end-user to relate to the global picture a minimap was implemented (figure 5). This minimap shows which nodes are currently on screen, and the position in the graph. Furthermore the user can click on the minimap to go to this specific location.

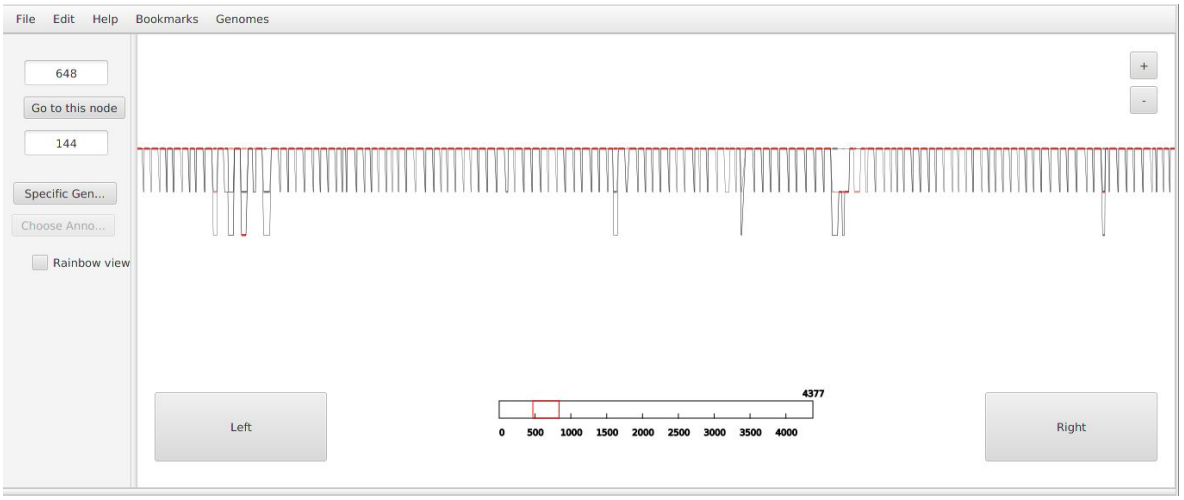

Figure 5. Minimap

Annotations search

The user is able to search through all the genomes on substring, the jump the selected genome still has to be implemented.

Special section on interaction design (development of the HCI module)

will be written in the weekend, after we've finished our HCI

Introduction

This evaluation will be used to check whether our created application is useable. In a questionnaire there will be some tasks described, which a person without former knowledge about our program or its contents should be able to execute. If we notice some tasks are harder than expected or can't be executed at all, we can evaluate this and change our application accordingly.

Method
- (option design of the study. For example if you have multiple conditions)
- participants
 Jips Mother
- measures used
 For each task the number of clicks will be counted as well as the time per task. Each task will have a predefined number of clicks and an estimated time (experienced - expected - max time).
- procedure
TODO: Create a questionnaire for the user.
- analyses
- etc.

Results
- figures, tables, average, result statistical analyses
- topics identified from for example qualitative analysis
- problems identified
- etc.

Discussion/conclusion
- answering the initial question
- major finding and possible interpretation
- suggestions for improvements
- limitations of the evaluation
- etc.

Appendix
- raw data
- questionnaires used
- task instructions
- etc.

Evaluation of the functional modules and the product in its entirety, including the failure analysis

For this purpose, an evaluation of the functionalities performed using a well-justified method needs to be presented, as well as a failure analysis – where the product does not perform as needed.

will be written in the weekend, after we've finished our HCI

Outlook

As Leonardo da Vinci once famously told "Art is never finished, only abandoned." . As is the case with our software product. Software is never finished and never perfect. This means that there always is a lot of room for improvement. Our software can be improved in regards to features, robustness, speed, and interaction.

Features for improvement

When we look at the features of our program a couple of them can be improved, the most notable being annotation, settings, and semantic zooming. When we look at annotations a lot can be improved. In our current version we can't jump to specific annotations, annotations stack on top of each other which sometimes causes a weird visual layout. Another feature we neglected are settings, we have virtually no setting options for the end-user. We could add render size, rendering start position, color palette, etc. This was not done because it does not really add anything to the program and will take up time which could be spent on building features. Another feature improvement might be semantic zooming. In the current state of the program we only use semantic zooming for snips. This feature could be expanded to handle other structures which occur frequently.

Robustness improvement

Not only features can be improved but the robustness of the program as well. We have three main issues right now with the robustness of our program; Panning, UI bugs, and Testing. The program sometimes randomly stops panning, this can be "fixed" by using bookmarks, go to node, or pressing on the minimap. This is not ideal and if this program is developed further this would be a high priority issue as it hinders the user experience considerably. Another robustness improvement lies within the UI. JavaFX has some weird bugs with fullscreen mode and disappearing panels, which can be annoying. However this is a very difficult bug fix as the main problem lies within the JavaFX library. We could however improve the robustness further by increasing our test coverage and test quality. Due to the nature of our program (A heavy UI based visualisation program) it is very difficult to test, but more can be done to make it easier to maintain in the future.

Speed improvement

The last aspect of improvement is speed. We have two bottlenecks which should be addressed in future updates. We have a go to node function which is the underlying base for bookmarks and the minimap. The issue lies in the fact that this function is O(n) while it could be O(log(n)) using a divide and conquer tactic. The second bottleneck we have is the cold load. Due to the program growing over time more has to be parsed and saved. This causes our biggest file Tomato to have a cold load of 10 minutes. This is too long, but very difficult to improve. We could make a switch to relational databases, but this option would need to be researched thoroughly as a lot of the underlying code base would need to be changed.

Interaction improvement

will be written after test data is in.

Bibiography

Matuszewski C., Schönfeld R., Molitor P. (1999) Using Sifting for k-Layer Straightline Crossing Minimization. In: Kratochvíyl J. (eds) Graph Drawing. GD 1999. Lecture Notes in Computer Science, vol 1731. Springer, Berlin, Heidelberg.

Patarasuk, P. (2004) Crossing Reduction for Layered Hierarchical Graph Drawing (master thesis). Retrieved from https://fsu.digital.flvc.org/islandora/object/fsu:180382/datastream/PDF/view

Sugiyama, K. (2002). Graph drawing and applications for software and knowledge engineers. River Edge, NJ: World Scientific.

Tamassia, R. (2010) Hierarchical drawing algorithms. London: Chapman & Hall/CRC.

Appendix A

Delft University of Technology
ETHICS REVIEW CHECKLIST FOR HUMAN RESEARCH

This checklist should be completed for every research study that involves human participants and should be submitted before potential participants are approached to take part in your research study.

In this checklist we will ask for additional information if need be. Please attach this as an Annex to the application.

Please upload the documents (go to this page for instructions).

Thank you and please check our website for guidelines, forms, best practices, meeting dates of the HREC, etc.

I. Basic Data

Project title:	**PL – Dynamites & Butterflies**
Name(s) of researcher(s):	
Research period (planning)	**24-06-2017**
E-mail contact person	lboleij@student.tudelft.nl
Faculty/Dept.	**EEMCS**
Position researcher(s):[1]	**Student**
Name of supervisor (if applicable):	**Willem-Paul Brinkman**
Role of supervisor (if applicable):	**Grading**

II. A) Summary Research

Our research will involve a user test of testing the usability of our program by a person that doesn't know much about the context of the app. For our project, a usability test of only one person should be enough, as eliminating user unfriendly features is the most important thing, which will come to light by reviewing the one participant. Devices to be used are a simple laptop on which the subject will have to try to execute some predefined exercises in the application.

B) Risk assessment

We don't expect this research to involve any risks towards the subject. He will only have to perform some simple task involving a simple computer mouse and keyboard, which the subject uses on a daily basis.

III. Checklist

	Yes	No
1. Does the study involve participants who are particularly vulnerable or unable to give informed consent? (e.g., children, people with learning difficulties, patients, people receiving counselling, people living in care or nursing homes, people recruited through self-help groups).	☐	■
2. Are the participants, outside the context of the research, in a dependent or subordinate position to the investigator (such as own children or own students)?[2]	☐	■
3. Will it be necessary for participants to take part in the study without their knowledge and consent at the time? (e.g., covert observation of people in non-public places).	☐	■
4. Will the study involve actively deceiving the participants? (e.g., will participants be deliberately falsely informed, will information be withheld from them or will they be misled in such a way that they are likely to object or show unease when debriefed about the study).	☐	■
5. Will the study involve discussion or collection of information on sensitive topics? (e.g., sexual activity, drug use, mental health).	☐	■
6. Will drugs, placebos, or other substances (e.g., drinks, foods, food or drink constituents, dietary supplements) be administered to the study participants?	☐	■
7. Will blood or tissue samples be obtained from participants?	☐	■
8. Is pain or more than mild discomfort likely to result from the study?	☐	■
9. Does the study risk causing psychological stress or anxiety or other harm or negative consequences beyond that normally encountered by the participants in their life outside research?	☐	■
10. Will financial inducement (other than reasonable expenses and compensation for time) be offered to participants?	☐	■

Important:
if your answered 'yes' to any of the questions mentioned above, please submit the full application form to HREC
(see: HREC website for forms or examples).

[2] **Important note concerning questions 1 and 2.** Some intended studies involve research subjects who are particularly vulnerable or unable to give informed consent. Research involving participants who are in a dependent or unequal relationship with the researcher or research supervisor (e.g., the researcher's or research supervisor's students or staff) may also be regarded as a vulnerable group . If your study involves such participants, it is essential that you safeguard against possible adverse consequences of this situation (e.g., allowing a student's failure to complete their participation to your satisfaction to affect your evaluation of their coursework). This can be achieved by ensuring that participants remain anonymous to the individuals concerned (e.g., you do not seek names of students taking part in your study). If such safeguards are in place, or the

	Yes	No
11. Will the experiment collect and store videos, pictures, or other identifiable data of human subjects? [3]	☐	■
If "yes", are you sure you follow all requirements of the applicable data protection legislation? *(Please provide proof by sending us a copy of the informed consent form).*	☐	☐
12. Will the experiment involve the use of devices that are not 'CE' certified?	☐	■
Only If 'yes': continue with the following questions:		
• Was the device built in-house?	☐	☐
• Was it inspected by a safety expert at TU Delft? *(Please provide device report, see: HREC website)*	☐	☐
• If it was not built in house and not CE-certified, was it inspected by some other, qualified authority in safety and approved? *(Please provide records of the inspection).*	☐	☐
13. Has or will this research be submitted to a research ethics committee other than this one? (*if so, please provide details and a scan of the approval or submission, if available).*	☐	■

IV. Enclosures (tick if applicable)

- Full proposal (if 'yes' to any of the questions 1 until 10)
- Informed consent form (if 'yes' to question 11)
- Device report (if 'yes' to question 12)
- Approval other HREC-committee (if 'yes' to question 13)
- Any other information which might be relevant for decision making by HREC

V. Signature(s

Signature(s) of researcher(s)
Date: 21-06-2017

Signature research supervisor (if applicable)
Date: