Table 1: Revision History

Date	**Developer(s)**	**Change**
September 26, 2016	Eric Chaput	Created Draft 1
September 27, 2016	Kelvin Lin	Modified Team Meeting Plan for Draft 2
September 28, 2016	Kelvin Lin	Added Team Member Roles for Draft 2
September 28, 2016	Kelvin Lin	Added Git Workflow Plan for Draft 2
September 29, 2016	Eric Chaput	Added to Sections 3,6, and 7
September 30, 2016	Kelvin Lin, Eric Chaput, Jin Liu	Spelling and Grammar edits
November 4, 2016	Eric Chaput	Revisions based on TA comments.
December 6, 2016	Kelvin Lin	Added Reflection
December 7, 2016	Eric Chaput	Final Edits for Revision 1

SE 3XA3: Development Plan
GrateBox

Team 8, Grate
Kelvin Lin (linkk4)
Eric Chaput (chaputem)
Jin Liu (liu456)

Blue text indicates rev 1 updates

1 Abstract

Having stated the problem to be solved in the problem statement document, Grate will develop the Genetic Cars project to solve it. The following document details the development plan for Grate's Genetic Cars project..

2 Team Meeting Plan

2.1 Overview

Grate's meetings will resemble a modified version of daily scrum meetings; however, instead of holding daily scrum meetings, they will be held weekly. Grate will structure 3 meetings each week, with the first meeting requiring full attendance of all members, and the remaining 2 requiring members to attend as needed. The first meeting will resemble the daily scrum meeting, while the latter two meetings will resolve issues and difficulties mentioned in the former meeting.

2.2 Meeting Structure

Team meetings will be held three times per week: on Monday from 7:00PM to 8:00PM, on Wednesday from 8:30AM to 10:30AM, and on Fridays from 2:30PM to 4:30PM. On Mondays, the team meetings will be held in a meeting room at McMaster University. Kelvin Lin will be responsible for booking the meeting rooms, and releasing the meeting location 1 week before the meeting date. The meetings on Wednesday and Friday will be held in ITB 236.

The meetings on Mondays will be used to aggregate the achievements and difficulties of the previous week, and to plan for the upcoming week. Members

will be presented with the opportunity to discuss any achievements and difficulties they encountered while working on their tasks. A list of difficulties will be created; however, there will not be an in-depth investigation or resolution of the aforementioned difficulty. For each issue mentioned, a list of team members who are qualified to solve the problem will be created, and a member from that list will be chosen to investigate that problem. A meeting will be held on the subsequent Wednesday or Friday in order to discuss the resolution or further issues pertaining to that problem. The meetings on Monday will conclude with a list of tasks each member is responsible for in the upcoming week.

The meetings on Wednesdays and Fridays will be used to address specific decisions pertaining to the project deliverables. Topics will include making specific design decisions, resolving issues mentioned on Monday's meeting, resolving any issues discovered throughout the week, and addressing any intergroup problems. To account for in-lab activities, the meetings will be divided into sub-meetings, and members will only be required to attend meetings that concern aspects of the tasks that they are working on. Members can work on the in-lab activities when they are not in a meeting.

2.3 Components of Meetings

Key components of team meetings include the chair, the agenda, the scribe, and the meeting minutes.

2.3.1 The Chair

The chair will be responsible for organizing the topics to be discussed at the meeting and orchestrating the flow of the meeting. Before each meeting, the chair is responsible for creating an agenda for the meeting. During the meetings, the chair has an obligation to maintain a neutral front; the chair exert his authority to favour one side of an issue over the other side of the issue. However, the chair will have the authority to defer topics that threaten the overall structure of the meeting: if a topic exceeds the allotted time for discussion or if an unforeseen issue arises from a predetermined topic, the chair has the authority to defer the topic to a future meeting. The chair will not have the authority to veto decisions made by the team, nor will the chair's opinion be valued more than the other members. At the end of each meeting, the chair is responsible for establishing the next meeting date and making the appropriate reservations for a meeting space. Unless otherwise noted, Kelvin Lin will be the chair of the meetings.

2.3.2 The Agenda

The agenda is a written document to be written in LaTeX. It details the topics to be covered in an upcoming meeting. At a minimum, it must provide a numbered list of topics to be covered, as well as a brief summary of each topic.

The summary can be as simple as a point list. The agenda should be released to all team members before each meeting.

2.3.3 The Scribe

The scribe of the meeting will be responsible for taking appropriate notes at the meeting. Before each meeting, the scribe must prepare the appropriate equipment needed to take notes. This could include, but is not limited to, laptops, notebooks, pens, pencils, and paper. If possible, the scribe should obtain a copy of the agenda from the chair. During each meeting, the scribe has the obligation to keep accurate, unbiased notes that provide a faithful representation of the meeting. All events and decisions must be recorded in a manner so that they can be referenced in the future. Attendance, and future meeting dates should also be recorded. After each meeting, the scribe is responsible for formalizing the notes into meeting minutes. Unless otherwise noted, Jin Liu will be the scribe of the meetings.

2.3.4 The Meeting Minutes

The meeting minutes is a written document to be written in LaTeX. It is a formalization of the notes taken by the scribe during each meeting. The meeting minutes can be appended to the corresponding meeting agenda of the same date. At a minimum, the meeting minutes must contain three sections: a section to detail the key points of the conversation during each topic of discussion, a summary of the key decisions made during each meeting, and a list of tasks each member is responsible for before the next meeting. The meeting minutes should be released to all team members before the start of the next meeting.

2.4 Emergency Provisions

In the unlikely event of an emergency, impromptu meetings can be held on Thursdays 12:30PM to 1:30PM, at a location to be determined. All members are expected to be present at emergency meetings. Depending on the urgency of the meeting, an agenda for the meeting may not exist; however, the scribe will still be responsible for taking notes and releasing meeting minutes after the meeting. Topics at emergency meetings will only pertain to issues at-hand: that is, no long-term planning will be done at emergency meetings unless otherwise required by the deliverable. If required, additional meetings can be scheduled through the Facebook Messenger platform on an as-needed basis.

3 Team Communication Plan

3.1 Online Platforms

Grate will communicate with a variety of online platforms and in-person meetings. While the structure of in-person meetings is described above, the online

platforms used include Facebook, GitLab, and Telephone/Internet Phone.

3.1.1 Facebook

Facebook offers an instant messaging service called Messenger, which will serve as the team's primary form of online communication. Facebook offers message logging, real-time communication, and private messaging, all of which are convenient features for the team. All message logs are recorded by Facebook, which can be referenced in the future. Moreover, Facebook messages are delivered in real time, so members can stay updated on the status of the project. Private messaging will also allow members to discuss aspects of the project that would not interest the third member.

3.1.2 GitLab

GitLab is a repository management system that allows users to version their code. Grate will use GitLab to store the project and any relevant documentation. GitLab will be used because it provides a form of separation of concerns for each member, while allowing for concurrent access to the code. Members can add to the code repository without interference from other updates. GitLab also allows members to have local copies of the code for experimentation. Accordingly, GitLab will be used to manage documents used by Grate.

3.1.3 Phone

The phone provides real-time voice-to-voice communication between members. The phone will be used for emergency contact purposes. Such emergencies include power outages, last minute changes in plans, and last minute technical errors. The phone will be used for emergency purposes because it provides synchronized communication between the members: the communicating members are expected to respond as soon as communication ends from the other member.

4 Team Member Roles

Grate will have the following roles: team lead, technology expert, graphics expert, theory expert, qualitative testing expert, and quantitative testing expert. The assignment and descriptions of each role follow.

4.1 Assignment of Roles

The following table shows the assignment of roles in Grate.

Team Member	Role
Kelvin Lin	Team Lead, Technology Expert
Eric Chaput	Theory Expert, Qualitative Testing Expert
Jin Liu	Graphics Expert, Quantitative Testing Expert

Table 2: Grate Role Assignments

4.2 Description of Roles

4.2.1 Team Lead

The team lead is responsible for the overall management of the project. The team lead is responsible for ensuring that all deliverables are made to specification and on time. To accomplish this, the team lead must be sensitive to the member's skills and abilities, and partition tasks to allow all of the other members to succeed and to develop new skills. The team lead shall ensure that every member is aware of the tasks that need to be done, and what their role is on the team. The team lead is also responsible for facilitating communication to external parties by representing the voice of the team.

4.2.2 Technology Expert

The technology expert is responsible for maintaining current knowledge on general aspects of the technology used. The technology expert will serve as the first point of contact for any technical problems encountered with, but not limited to, Git, HTML5, JavaScript, LaTeX, GanttProject, Doxygen, and Karma. The technology expert will be in charge of general troubleshooting, and he will redirect members to the appropriate experts when required. Moreover, the technology expert will be responsible for researching new technologies that can be used for the project.

4.2.3 Graphics Expert

The graphics expert is responsible for maintaining current knowledge on the algorithms and technology needed to implement the graphics for the project. The graphics expert shall learn all tools necessary in order to create a graphical interface for the project, and he will be responsible for communicating this knowledge to other members of the team in a manner comprehensible by both members with and without prior technical knowledge. The graphics expert will also be responsible for experimenting with new graphical technologies, and spearheading the graphics-related implementation initiatives of the project.

4.2.4 Theory Expert

The theory expert will be responsible for maintaining current knowledge of the theory needed to comprehend the algorithms used in the project. The theory expert will be responsible for understanding theory behind the physics, learning

algorithm, and graphics. The theory expert will be responsible for finding any references to published text when needed. Moreover, the theory expert will be responsible for communicating this knowledge to other members of the team in a manner comprehensible by both members with and without prior technical knowledge.

4.2.5 Qualitative Testing Expert

The qualitative testing expert will be responsible for maintaining current knowledge of qualitative testing methodologies within the scope of the project. The qualitative testing expert will be responsible for leading any qualitative tests including but not limited to market research and observation, in-person and online surveys, discussion/focus groups, and code review sessions. Such tasks may involve partitioning tasks for each initiative and assigning the tasks to the appropriate member. Moreover, the qualitative testing expert will be responsible for reporting any results to the group and making recommendations based on the results to improve the projects.

4.2.6 Quantitative Testing Expert

The quantitative testing expert will be responsible for maintaining current knowledge of quantitative testing methodologies within the scope of the project. Before the implementation of the project, the quantitative testing expert shall familiarize himself with different tools used to test the agreed-upon technologies. The quantitative testing expert will be responsible for leading any quantitative testing initiatives including unit testing, integration testing, system testing, automated testing, and manual testing. Such tasks may involve partitioning tasks for each initiative and assigning the tasks to the appropriate member. Moreover, the quantitative testing expert will be responsible for reporting any results to the group and making recommendations based on the results to improve the project.

5 Git Workflow Plan

Grate will utilize a feature-branch workflow plan.

5.1 Feature-branch Overview

The feature-branch workflow branch is a Git workflow plan that utilizes different branches to implement different feature of the programs. Once a particular feature of the program is completed, its branch will be merged with the master branch. Labels will be used to tag submission files, while milestones will be used in order to signal any major revisions on the master branch.

5.2 Rationale

In selecting a Git workflow plan, Grate wanted a workflow that would provide maximum support for separation of concerns, while guaranteeing that there will always be a working version of the project easily assessable. Such workflow would simplify both the development, testing, and deployment efforts of the team. A high level of separations of concerns would allow multiple members to code concurrently without interference from code changes from other members. Members can also test their code using the master branch as a driver. This is a valid approach as any functional code on the master branch would have already been tested. The second requirement of the Team's desired workflow plan guarantees this by specifying that only functional versions of the project are to be committed to the master branch.

The feature-branch workflow plan fully satisfies the requirements mentioned above. It supports separation of concerns by placing each feature of the program on a separate branch. Each branch will stem from a node on the master branch, allowing each branch to have an independent copy of the previously functional code. This will allow members to write new features in isolation, without having to account for continuous changes made by other members. The master branch can also serve as a driver for unit testing, and for regression testing. Accordingly, the feature-branch workflow was chosen.

6 Proof of Concept Demonstration Plan

The proof of concept demonstration will consist of three independent components that resemble simplified versions of the final product: the car model, the genetic algorithm (GA) module, and the graphics simulation.

6.1 The Car Model

The Car Model will be a JavaScript module that demonstrates two things: the car is representable as a data structure, and the Box2D API can be integrated into the project.

6.1.1 The Car

The car in this project is composed of 8 vectors attached to a central point. The tips of the vectors are connected to form triangles. Each car contains 8 vectors. In addition, each car can have up to 8 wheels. Each wheel has a radius, an axle angle, and a vertex. Two wheels can lie on the same vertex. Each car will also have mass, which will be a randomly generated value between two predetermined numbers.

6.1.2 The Deliverable

The Car Model will likely be modelled as a JSON object, or a customized equivalent, with properties such as the coordinates for each of its 8 vectors; the vertex, axle angle, and radius of each of its 8 wheels; a boolean for each of its wheels to indicate whether the wheel exists. A script to test the JSON model will also be created. This script will access each field of the car and output its value in text to the console. The script will also use the Box2D API to create the corresponding vectors for the car, and if it is successful, then it will print "true" to the console.

6.1.3 The Risks

The Box2D API poses the most significant risk for the car model. The Box2D API defines the car entity in terms that can be used with many physics equations, which is important for calculating the fitness function of the car. In the event that the Box2D API prooves to be infeasible for Grate, alternate arrangemets will have to be made in order to complete the project: the team will resort to using basic kinematics equations to calculate the fitness function instead of using the API. A possible drawback to this approach would be that the members of Grate is generally unfamiliar with Newtonian mechanics, so external assistance would be required. Tentatively, the team plans to consult with Dr. Spencer Smith during his office hours should this assistance be required.

6.2 The Genetic Algorithm

The purpose for implementing the GA is to demonstrate its practicality to the project. In order for this portion to succeed, it must be able to select for the top n cars given an array of cars, n, and fitness scores as input; crossover the genes of these cars; and mutate the genes of the offspring.

6.2.1 Genetic Algorithm Overview

GAs attempt to find near-optimal solutions to problems with incomplete or imperfect information by emulating the process of natural solution. In this project, the GA will be used in order to find the best car in terms of the distance driven on an predetermined road. First, the GA will select for the top n cars based on their fitness scores. Then, the GA will use the fields of the car as genes, and swapping them in order to simulate reproduction. This project will do this by randomly selecting two fields and two cars, and swapping those two fields such that the first car will have the two fields from the second car and the second car will have two fields from the first car. Finally, to finish creating the offsprings, the GA will randomly change some of the fields at a frequency based on a mutation likelihood between 0% and 100%.

6.2.2 The Deliverable

The deliverable will consist of a JavaScript file that consists of at least three functions: selectNextGeneration(cars:Car[], fitnessScores: Integer, n: Integer), crossoverOffsprings(cars: Car[], population: Integer, n:Integer), and mutateOffsprings(cars: Car[], mutationFactor: Integer, n: Integer). The selectNextGeneration function will take an array of cars and their fitness scores, and return an array containing the top n cars. The crossoverOffsprings function will take an array of cars, and randomly select parent cars (from the first n element in the cars array) to cross over their genes, such that the number of cars in the cars array equals the population. The mutateOffsprings function will randomly mutate the fields of the offsprings by a given mutation factor.

This file will be demonstrated using another script with hardcoded values for cars. The script will first run the selectNextGeneration function, and print the fields of the top n cars to the console. Then, the script will crossover the top n cars using the crossoverOffsprings function, and print their fields to the console, as well as which fields and cars were used to make the swap. Finally, the mutateOffsprings method will be used to mutate the cars and then the fields of the cars that mutated as well as their previous fields on separate lines.

6.2.3 The Risks

The adaptation of algorithm itself presents a significant risk for this project. Any assumptions made in the original algorithm will carry over to the project where those assumptions may not hold to be true. Furhtermore, if any errors exist in the GA code, the final result will be negatively effected. To avoid this risk, the original algorithm will be tested in an isolated enviroment to ensure its general accuracy. The coded algorithm will also be tested in isolation of the program's other components to ensure it is properly implemented in Javascript. This is feasible due to Grate's object oriented programming approach to this project.

6.3 The Graphics

The purpose of the graphics demonstration is to show competency in the technologies required to create the graphical interface for the project. The final deliverable will consist of a simple 3-vector car moving horizontally across the screen over a dynamically generated road, with a camera centred on the car.

6.3.1 Graphics Overview

The graphics for this project consists of multiple cars displayed on the screen, moving horizontally over a road, with the camera focused on the fastest car in the population. The user interface also allows users to modify certain parameters of the GA, and view the improvement of the cars over several generations.

6.3.2 The Deliverable

The deliverable for the proof of concept demonstration will be a simplified version of the final user interface. It will consist of a HTML5 canvas with a dynamically generated line representing the road near the bottom of the canvas. A random car will be generated with 3 vectors (1 triangle) and 2 wheels (2 circles). The car will move horizontally across the screen indefinitely. A camera will also focus on the car, so that the center of the car will always be at the center of the screen.

6.3.3 The Risks

The implementation of the Javascript graphics language will present a significant risk for this project. Grate has more experience with C++ and Java graphics packages, and it is possible that the adaptation of Javascript will prove more difficult then initially believed. Grate's graphics and technology experts have arranged to minimize this risk by educating themselves on Javascript's graphics functionality extensively through online resources and peers familiar with Javascript.

7 Technology

This project will use technology that all group members are already familiar with on a basic level. The project itself will be coded in Java Script, however all group members understand that Java may be used if JavaScript proves too difficult to use for this project. All group members are familiar with both of these languages and no further learning will be required. Grate's graphics expert is also adept at creating graphics in JavaScript. Written reports and documents will be generated through Latex, a technology being taught in the 3xa3 course, and documents and reports will be shared through Git, another technology being taught in the 3xa3 course. The technology expert will ensure that all group members are aware of how to use these technologies. Testing will be aided by the use of Javas built in unit test functionality. Dox will serve as the primary document generator, and QUnit will serve as the means of automated testing.

7.1 JavaScript

JavaScript as a programming language is used for web applications. JavaScript is the ideal programming language for this project. Member's of Grate is familiar with JavaScript, and Grate's graphics expert believes that JavScript will work best for his role. The final project will also closely resemble a standard web page in design. Finally, since the original basis for the project is also written in JavaScript, the original source code code can be more easily used for inspiration and troubleshooting.

7.2 LaTeX

LaTeXwill be used to generate documents for this project. LaTeX's ability for multiple team members to effect changes simultaneously and its versatility across platforms makes it ideal for this project. As of September 28 2016 all team members have acquired a working knowledge of how LaTeXfunctions, and it is believed that no further education will be required. If any such education is necessary the TAs and the Latex tutorial found online will be Grate's go-to instructors.

7.3 GitLab

GitLab will be used to share and store documents for this project. More information on GitLab and how it will be used for this project can be found in sections 2.1.2 and in the Git Workflow Plan found in section 4. All Grate members are familiar with GitLab and how it functions.

7.4 Dox

Dox will be used to generate documents. Dox is a JavaScript documentation generator written with Node available on GitHub. It generates a JSON representation from the comments in the code, which can be passed to a template in order to generate a website similar to JavaDocs. Its well documented nature makes it a good choice for this project. The original source of Dox can be found at https://github.com/tj/dox.

7.5 QUnit

QUnit will be used for automated testing for this project. Its similarity to JUnit, an automated testing software for Java, means that it will be easy to learn, and its cross platform flexibility means that it is unlikely to cause technical problems. The original source of QUnit can be found at https://gunitjs.com

7.6 BootStrap

An HTML extension that helps with web design. Will be used to better the UI of our project. Completely open source. Mostly self explanatory use.

7.7 JQuery

A JavaScript extension that makes it easier to create buttons. This will be used for the pause button in our final product. Use is self explanatory for the most part. An open source technology.

8 Coding Style

Code will be styled primarily as has been taught by the software engineering program at McMaster University. Grate will undertake an object oriented programming approach to achieve maximum modularity and information hiding. The team shall also ensure that the code meets several aesthetic guidelines outlined here. All code is to be well commented, with comments being placed before the code. Comments are to be written with the understanding that they should be concise and easy to comprehend by outside parties. To satisify these requirements, Mozilla Developer Network's coding style will be adopted by the team.

8.1 Mozilla Developer Network

Mozilla Developer Network will be Grate's default coding style. It contains subsections specifically for JavaScript, the programming language of choice for this project. The guidelines for this coding style can be found at https://developer.mozilla.org/en-US/docs/Mozilla/Developer_guide/Coding_Style.

9 Project Schedule

Please find the project schedule at
https://gitlab.cas.mcmaster.ca/linkk4/GrateBox/tree/master/ProjectSchedule.

10 Project Review

The past four months have been successful for Grate. Grate accomplished all of the goals it specified in the beginning with a high degree of success. Grate was able to re-implement Genetic Cars by rednuht.org while creating extensive documentation on the project following standard software engineering principles.

The members of Grate were able to experience first-hand the thrills and challenges of implementing a medium-scale project, and this experience allowed all members to grow in significant ways: both technically and managerially.

From the beginning, a key lesson Grate learned was to start coding early, but not become attached to the code that is written. The proof of concept demonstration proved effective in showing Grate that proper software design techniques need to be used in order to build an extensible system. Initially, the proof of concept was built in a cut-and-paste fashion, reusing parts of code that achieved some desired function. However, it was through this demonstration that Grate realized that it would be extremely challenging to modify this code in the future. The proof of concept helped set Grate on a road to using software design principles in order to build a strong and understandable system.

Moreover, creating the different documents for the project allowed the members of Grate to build a project that is bigger than what any individual member could have built. By writing complete and relevant documents, the individual

members of Grate were able to communicate asynchronously with other members about the tasks that they were working on. This allowed for members to build off on each other's code before the other person's code was complete. Of course, this was effective partially because of each member's trust in each other, and the common desire to succeed and build an amazing product in this course.

Macroscopically, the iterative process that Grate used was very effective in developing software project management skills. The Gantt chart was used extensively in the software development process as a guideline for each member when direction was needed. The use of Gitlab Issue Tracking was also helpful in decomposing tasks that each member needed to do, as well as communicating with other members about the status of each member's work. The use of version control and branching allowed for multiple copies of the project to exist on different machines at the same time - members did not have to wait for other members to finish their section of the code before starting their own. These tools allowed each member of Grate to refine their teamwork, communication, and time management skills, which benefited the team as a whole.

Nevertheless, GrateBox is still in its early stages of development, and there are still tasks Grate has left incomplete at the time of writing. On the technical side, there are several different types of genetic algorithms that GrateBox could be made to support. Moreover, support for more parameters can be made for the user to customize.

With respect to software project management, the use of PERT charts, agile methodology, and continuous integration could be implemented. The use of PERT chart would extend the effectiveness of the Gantt chart, so that each member knows when they have slack time and when they are on the critical path. Agile methodology could perhaps expiate the process of software development through more frequent, smaller iterations. Continuous integration would allow the software to be tested every time a build is created. By implementing all of these features, GrateBox would be a more powerful application to teach students and professionals the power of Genetic Algorithms.

Ultimately, the members of Grate started this project with no prior experience with software project management. However, after going through several iterations of the design process, every member became adept at effectively contributing to each step of the process. By using different software management tools, GrateBox became what Grate considers a successful application in under four months.