COMP-361 Software Engineering Project
Flash Point

v1.0

Flash Point is a cooperative multiplayer board game designed by *Kevin Lanzing* and published by Indie Boards and Cards. The objective is for players to work together to rescue people and animals from a burning building before it collapses. The original game was released in 2011 and was since extended with many expansions (*2nd Story*, *Urban Structures*, *Extreme Danger*, *Dangerous Waters*, *Veteran & Rescue Dog*, *Honor & Duty*, *Tragic Events*), which can also be combined.

You are to implement a multiplayer, distributed version of *Flash Point* according to the rules of the base game.

1 Game Rules

The rules in the game manual describe the game play in detail. The game manual for the base game is provied for download in PDF format on MyCourses. You are to implement the game exactly as specified in the base rules, including the family game play and the experienced game play. You can also find it on the official website:

- http://www.indieboardsandcards.com/index.php/games/fp-fire-rescue/

There is also a video explaining the family game play here:

- https://www.youtube.com/watch?v=eRAuAt0XBoE

A video explaining the experienced game play can be found here:

- https://www.youtube.com/watch?v=aPs8D47iitM.

Extending the Game Rules

You are allowed to extend the game by coming up with additional game features, e.g., new roles, special events, different houses, etc. In case you want to do that, we require you to document the game rule changes/extensions precisely, discuss them with us to ensure that the changes you suggest do not simplify the game design and implementation, and commit to them before the end of the first semester. Each of your project submissions has to include the changes to the rules so we can consider them during marking. Failure to do so will result in marks taken off.

2 Additional Requirements

The game manuals describe the rules for the board game. Since you are implementing a computerized version of the game, you must in addition to the game rules provide certain computer-specific functionality in order to get full marks for your implementation.

2.1 User Interface

In your implementation you have to design your own user interface. Of course you need to visualize all the elements of the board game, but your user interface also needs to allow the player to trigger all the functionality needed to play the game. Try to create a user interface that is as intuitive as possible. Exploit the fact that you are running on a computer to augment the user interface with features that provide guidance for the player, e.g., context-sensitive help. Also, think of UI elements that are needed to support the multiplayer experience, e.g., messaging, etc.

As the minimum, the software must help a player to complete his turn correctly and in an efficient way. For example, a good UI should not allow a player to make incorrect moves. Also, since this is a collaborative game, it is important to provide ways so that the players can together elaborate a good firefighting strategy.

2.2 Distribution and Game Server

The application must make it possible for (at least) three players (maximally 5) to play against each other over a network. This means that somehow, the players must be able to set up a connection between at least three devices (see below), agree on playing with each other and on which game variant, and then play their moves turn by turn until the game is over. To facilitate players to setup a game, you should implement a game lobby that allows players to connect with other players that are currently online.

2.3 Implementation Platform

You can use any implementation platform you like, provided that the programming language is object-oriented. However, we will provide technical support for students using Java. Also, we developed a Java-based game programming framework called *Minueto*[1], that takes care of efficient rendering of graphics (off-screen drawing, transparency, hardware acceleration, etc.). That way, students who do not know much about Java graphics will not have to waste time learning these concepts, but can focus on the more important functional part of the application. Minueto also does some basic user input handling and provides sound output.

[1] http://minueto.cs.mcgill.ca/

You are allowed to make use of external libraries/frameworks, e.g., for networking, graphics, concurrency-related, databases etc. However, you are **not** allowed to reuse/copy code from existing implementations of Flash Point that you find on the Internet to implement functionality which is required for the project.

Finally, please also keep in mind that the demonstrations and the acceptance test take place in the Trottier building and that the wireless network of McGill can provide challenges when setting up a network between several devices. You can, of course, bring your devices (smartphones, tablets, game consoles, etc.). Some students in the past have even brought their router to set up a local network to circumvent these challenges.

2.4 Different Boards

During setup, the players should be able to choose to play with one of the 2 original houses/boards, or to generate new random ones.

2.5 Saving and Loading

The application must allow a player to save a game. Later, it should be possible to load a previously saved game and continue playing from there.

2.6 Chat

Since Flash Point is a cooperative game, the application must include at least a chat function that helps the players communicate to be able to come up with a strategy and discuss their actions among each other. Also, try to think of innovative ways on how the application can help with this, since discussion can be fairly elaborate among the players on what the overall strategy is and which actions a player should perform.

3 Project Milestones

The project is to be done in groups composed of 6 students. Groups have to be formed by September 17th. The workload has to be spread equally among all group members and each team member has to contribute equally to the success of the project. Usually, this works best if all group members have similar ambitions for the project. We recommend therefore that you discuss this openly within the group before committing to be part of the team. During the year, if you are experiencing problems within your group (e.g., communication problems, or you are feeling that the project workload is not equally shared among team members), try to resolve it within your team first. If this does not help, then *don't* wait, but talk to the instructors or TA about it *as soon as possible*. The earlier we know about a potential problem the better, and then we can help to solve it!

The final grade of the project (65% of total grade) is composed of:

- 3% for the user interface sketch (October 1st)
- 5% for the requirement elicitation document (use case model) (October 17th)
- 10% for the requirements specification document (concept, environment, operation and protocol model) (November 20th)
- 5% for the user interface demo (last week of November)
- 10% for the design document (design and interaction model) (February 11th)
- 10% for the demo (March)
- 22% for the acceptance test (April)

5% of each assignment above is attributed for having organized your work among your group on Basecamp and/or GitHub (more information below). Details for each required hand-in are going to be announced separately, but below you can find a brief summary of each milestone so you can plan accordingly.

3.1 User Interface Sketch

Good user interfaces should prohibit the user from making "mistakes", such as unauthorized or useless moves. An intuitive user interface should also assist players whenever possible, for instance by highlighting possible positions for the player's firefighter.

You should hand in a sketch of the main screens that are presented to the players of the game. It should show how the player interacts with the game, how she sees the board when playing the game, how she communicates with the other team members, how she chooses actions, moves, extinguishes fire, etc., and how she accesses all other required game functions (e.g., saving the game).

3.2 Use Case Model

This hand-in comprises the use case model (requirements elicitation document), that clearly specifies the use cases pertinent for a player and her interaction with the system.

3.3 Requirements Document

This hand-in comprises requirements specification models that clearly specify the functionalities that your game implementation needs to provide. They are derived from the use case model and include a concept model, an environment model, a protocol model and an operation model.

3.4 User Interface Demo

For this milestone you will have to demonstrate a running user interface prototype, i.e., a user interface for the Flash Point game that shows how the players see the game board and how they will trigger some of the game's functionality. The game rules do not have to be implemented for this demo. The exact requirements will be communicated in a separate hand out.

3.5 Design Document

This hand-in comprises design models that provide a detailed blueprint of the structure (i.e., classes) and behaviour (i.e., methods) of your application that relate to the implementation of the game and its rules (the business logic). Note that the design models will not cover classes and methods that are needed to handle graphics and network communication.

3.6 Demo

After spring break you will be asked to give a first demonstration of your product. You will be given a list of minimal functionality (covering the fundamental aspects ot the game, such as some game rules, some GUI and network connectivity) that you must show during the demo. 75% of the grade for the demo is based on the correct implementation of the required functionality, and that you can run the demonstration without visible errors/bugs/crashes. 5% is attributed for organizing your work on Basecamp and/or GitHub (see below). The remaining 20% are attributed as follows: 5% for the quality of the demo (i.e., presentation/explanation of what is happening), and 15% for additional functionality of the game that you can already demonstrate, but that was not required for the demo.

3.7 Maintenance Phase

In the week of the demo, we will hand out some slight changes/additions to the game rules. This simulates "real-life" software development, in which the application requirements are often subject to change during the development of an application. In order to prepare for this phase, try to come up with a flexible design and write structured, modular, extensible code.

3.8 Acceptance Test

In the last week of classes, your application will have to pass the "acceptance test". During the test, the group of graders will play your game, looking for implementation of game functionality and rules and any bugs/glitches or violations of the game rules. We run the show, and you basically just watch.

75% of the grade for the acceptance test is based on the correct implementation of the game rules (i.e., if your application fulfills all requirements explained in the game manuals and the additional ones listed in this document). 5% are

again attributed for group organization within Basecamp and/or GitHub (see below). If there are no errors/bugs/crashes during the acceptance test and you organized your work, you will get at least an A- (80). The remaining 20% are attributed for an elegant/intuitive user interface, or other extra features. For instance, you could get extra points by having a 3D user interface, or by adding additional fire complications, extensions to the game rules, or by providing player statistics in the game lobby, tournaments, etc. Note that although adding new rules to the game is fun, 75% of your grade is solely based on whether or not you implemented the required game rules exactly as specified. Hence focus on that first :)

In order to prepare for the acceptance test, please keep in mind that it takes a long time to "debug" an application of this size. Even under the assumption that you do extensive unit testing during development, we suggest to plan at least two weeks for final adjustments.

Some Suggestions

Here's a list of some simple suggestions you should keep in mind during the development:

- Strive to have a good group dynamic, keep everyone in the group in a "good mood".
- Split your work in smaller chunks/tasks and assign each to your group members.
- Have regular group meetings (e.g., once per week) to consolidate your work.
- Start thinking about your user interface, architecture, implementation platform, technologies, networking etc. very early, learn and test them out.
- Start implementation early.
- First strive for a simple, correct implementation. Later, if there is enough time, add more sophistication.
- Always keep the deadlines in mind. Organize and schedule your work at the beginning of each milestone and keep up to date with everyone's progress.
- Test your setup in Trottier at least once before the demo & acceptance test to make sure everything there works (e.g., the networking).
- Do not make big changes on the day that precedes the demo or the acceptance test (or else be sure to have a functional backup copy).
- Testing takes time!
- Plan for the unpredictable!

Penalties

While the project milestones receive a grade for the group as a whole, individual adjustments may be made under certain circumstances. These include, but are not limited to:

- not contributing to an assignment/project milestone results in a mark of 0
- failure to attend your team's project demo or acceptance test without consent from the instructors will result in a loss of 10% of the total mark for that milestone

4 Project Organization

You are required to organize all of your team's work on Basecamp and assign tasks to every member, and keep it updated. For each hand-in/assignment, 5% of the total mark will be attributed for having organized your work on Basecamp. We will provide you with a team on Basecamp within our course organization once the teams have been formed.

Simultaneously, you need to use GitHub for managing your source code (at least) in order to coordinate work on your code base among your team, but you are welcome to put your other documents under version control as well. During implementation, you are required to manage issues (tasks) for the development on GitHub and make use of pull requests. Each group will get their own private repository within our course organization.

Changelog

- v1.0: Initial version.