CCC Project 2015: Bomberman

ETH Zurich

Hand-out: 18.2.2015
Team registration: 25.2.2015
Final deadline: 3.5.2015
Project assistant: Mischael Schill (mischael.schill@inf.ethz.ch)
Mailing list: se-ccc@lists.inf.ethz.ch

1 Synopsis

The goal of this project is to apply a high-level concurrency model to a practical example: a networked Bomberman [3] game with a client-server model. In this case, the concurrency model is SCOOP, which will be presented in lecture 6.

2 Game Description

The following sections describe how the game should work.

2.1 Starting up

1. The player starts the game client and enters the connection information (hostname/ip-address and port) of the server he or she wants to play on as well as his/her desired username.
2. The client establishes a connection and sends a hello message to the server.
3. The server checks whether there is already a player with the given name connected, and sends either a welcome message or a duplicate name error message, causing the client to start over with a hello message and another name.
4. For every already started game, and every game started after the connection is established and the welcome has succeeded, the server sends a message with the game information.
5. The client can then either disconnect, create a game or join a game.

2.2 Creating a game

1. The player clicks on create game and is presented a dialog with several options.
2. The player has to give a (unique) name for the game, and a map file.
3. The clients send this information to the server.
4. The server checks for validity of the information and either sends the game information to all clients or returns an appropriate error message.

2.3 Joining a game

1. The player clicks on a game and then on join.
2. The clients sends a join request to the server.
3. If the game is still open, the server sends a success message and otherwise an error message.
4. If all positions of the game are filled, the server sends a message with a countdown time to the clients whose players joined the indicating that the game will start after the given amount of seconds. It also sends a message to all other clients that the game is no longer available.

2.4 Playing a game

The players steer Bombermen. The goal is to blow up all other Bombermen, notably without getting blown up. A player can move his/her Bomberman with the arrow keys; every keypress corresponds to a one-tile movement of the map. The map consists of two basic types of tiles: walls and floors. Walls are impenetrable. A floor can hold either nothing, a bomb, a crate, an item or any number of Bombermen. Moving onto a tile with a crate or a bomb is impossible, whereas moving on a tile with an item causes the Bomberman to pick it up, thereby removing it from the playing field.

Every Bomberman can drop one or more bombs, but only one per tile. By default, the Bomberman can only drop one bomb and not another until the first has exploded, but note that there is a type of item that allows to have more than one active bomb. When a bomb is dropped, its time starts ticking and after a certain time, by default 3 seconds, it explodes.

A bomb explosion has a certain range, by default 1. It covers, within its range, all tiles north, south, west and east of the tile it was dropped on with fire unless there is a wall or a crate in the way. If a crate is in the way, it is destroyed, possibly dropping an item. However, the crate still blocks the fire, so tiles behind the crate are not affected. If a bomb is affected by another bomb it immediately explodes, forming a chain reaction. One type of item increases the range of bombs, not including the already dropped bombs.

It is not possible to move on tiles containing bombs unless the Bomberman is equipped with an item called Kicking Boots. They allow the Bomberman to kick bombs in the walking directions all the way to the opposite wall. If there is no space to kick the bomb to, even a Bomberman with Kicking Boots cannot move onto the tile.

All actions (walking, dropping bombs) require a certain time to be completed. The server has to check that no other action is performed until this time has elapsed. The last type of item decreases this time, so it effectively speeds up the Bomberman.

The game is finished if at most one Bomberman is alive and there are no more bombs active, upon which the last Bomberman is declared the winner. It is possible that a game has no winner.

If a client disconnects during a game, its Bomberman drops dead, but all its remaining bombs continue to tick.

On the technical side: When a player presses a key to move or drop a bomb, the clients sends an appropriate message to the server. The server checks for validity of the action and then sends to all clients the result of the action, who then update their state appropriately. If the action was not valid, the server does nothing. It is up to the implementer whether a message within the action time windows is either dropped or enqueued, or a combination of these two options.

3 Environment

A VirtualBox image is provided on the course website [1] that has the necessary tools preinstalled. Both username and password are set to ccc by default. The root password is also set to ccc.

3.1 Setting up the virtual machine

1. Install VirtualBox from [2]
2. Download the Virtual Machine from `http://se.inf.ethz.ch/courses/2015a_spring/ccc/vm/CCC.zip`
3. Unpack the Virtual Machine
4. Start Virtual Box
5. Add the VM through Machine → Add ...
6. Start the virtual machine
7. Log in with username and password ccc
8. Start a terminal and type `setup.sh`
9. The VM is now set up for working on the project

3.2 Home directory structure

Desktop Contains the files displayed on the Desktop

Project The root directory of the project

- **Sources** The working copy of the team's Subversion repository
  - **bomberman_client** The client project files
  - **bomberman_server** The server project files
  - **bomberman_shared** The project files shared by the former two
  - **report** Where the report should be added (LaTeX preferred, otherwise PDF)
- **Libraries** Contains some non-standard libraries

Workspace Contains two pre-configured EiffelStudio SCOOP projects that can be used for exercises

3.3 Snapshots

The system is configured to make hourly snapshots of the home directory. It retains the last 24 snapshots. The snapshots are available from `/home/.snapshot`. **Do not rely solely on these snapshots as your backup!**

3.4 Important programs

The task bar contains five icons:

1. Terminology – a terminal
2. EiffelStudio
3. a shortcut for updating the libraries
4. rapidSVN – a graphical Subversion client
5. TexMaker – a LaTeX editor

3.5 Installing additional programs

The hard disk space for programs is limited to keep the VM file small. A full installation of TeXlive is therefore not possible. If you want a VM that has LaTeX installed, ask the project assistant for a variation of the VM with more space for programs and an installed version of LaTeX. The free space for the home directory should be large enough.

3.6 Firewall

The firewall is configured to allow access to the ports 143, 1143 and 1144. Since port 143 can only be bound by root, it is recommended to use 1143. Secure Shell (SSH) is both disabled and blocked since the password is too simple to guess. Enabling of SSH is only recommended if both the password of ccc and root is changed first. The VM is configured to forward these ports from the local machine to the VM if you want to test the server from outside of the VM.

3.7 VM configuration

The VM is configured for 512MB of RAM and one core. If your computer has more cores and/or more memory, you might want to change these parameters to speed up the VM.

4 Deliverables

Delivery is done through the team's Subversion repository. By default, the last commit before the deadline is considered. If a previous commit should be considered, please send a message to the project assistant.

4.1 Report (10 points)

It is expected that the program is well documented in the report. The report values up to 10 points and needs to encompass the following:

- Architecture description and the design decisions with a focus on processor placement (6 points)
- How SCOOP helped or hindered the implementation (2 points)
- Readability, structure, grammar (2 points)

4.2 Program (40 points)

It is expected that the program code is readable and documented. Poorly written code hampers the assistants' ability to award points if the program does not work as expected.

5 Programming Milestones

The program code for the following deadlines has to be on the subversion server by 23:59 on the given date.

5.1 Team registration (1 point) – Deadline: 25.2.

Register your team as described in section 6. Timely registration is rewarded with 1 point.

5.2 Basics (3 points) – Deadline: 15.3.

Implement the 6 features that are marked using `TODO: BASIC FUNCTIONALITY`. When all these features are implemented, the clients can connect to the server and receive a welcome message and an empty list of available games. The main goal of this deadline is to make you familiar with Eiffel and some of the classes used in the program. No SCOOP-specific knowledge is required.

- Every correct feature implementation: 0.5 points

5.3 Game handling and movement (8 points) – Deadline: 2.4.

For this deadline, the players can create and join games as well as move around the battlefield, restricted by walls and crates. Not part of this deadline is the placement of bombs and subsequent functionality like items etc.

- Creating games: 2 points
- Joining games: 2 points
- Movement: 4 points

5.4 Basic gameplay (18 points) – Deadline: 19.4.

This deadline completes the basic game: bombs, items, crates and win/lose.

There are four basic items available:

- Bomb range increase: increases the range of all future bombs by this player by one (default: 1)
- Speed increase: decreases the duration of player actions (movement, placing bombs) by 10 % (default: 0.5s)
- More bombs: increases the number of bombs a player can set concurrently (default: 1)
- Kicking Boots: allow the player to kick bombs

The Bomb Kicker Boots allow a bomb to be kicked in the opposite direction by moving onto its field, but only if the way is clear.

- Placing bombs: 2 point
- Bombs exploding: 3 points
- Killing bombermen: 3 points
- Crates: 3 points
- Destroy items: 1 point

- Basic items: 3 points
- Kicking boots: 2 points
- Determine winner/losers: 1 point

5.5 Final submission (8 points) – Deadline: 3.5.

For the final submission, additional gameplay and architectural features are required. The goal is to remove concurrency bottlenecks from the game as well as add more spice to the game.

The basic battlefield is a monolithic construct that holds all the positions of all the objects and actors in the game. Since only one processor can access it at a time, it's architecture limits the scalability of the game. One way to remove this bottleneck is to segment the map into sub-maps. A good balance has to be found to both minimize the amount of locking as well as maximize the amount of concurrency. It is possible to only require at most 3 locks per request to the map and still have a minimal size of the sub-maps.

Since a client cannot show a large map at once, it is prudent to only send relevant information to the client. The viewport of a client should be 16 by 16 squares around the main character. Only information about what is going on inside these squares should be sent to the client. Of course, off-viewport explosions that range into the viewport should be considered as well. When a player moves, the server sends information about the newly visible parts of the map.

- No concurrency bottlenecks (general): 2 points
- Segmented battlefield: 1 point
- Optimal sub-map size and segmentation: 1 point
- Large maps in client: 2 points
- Viewport limited communication: 2 points

5.6 Game presentation / Tournament (2 points) – 6.5.

In the seminar slot on the 6.5., a great tournament will be hosted. All teams that show up receive two points. During the tournament, all finished games are first presented by their team and then played by all. The winners receive a small prize.

6 Teams

You can work in teams of up to three persons. In order to register your team, send a message with your chosen team name, the nethz-usernames and, if not username@student.ethz.ch, e-mail addresses of the members to the project assistant by **25.2.2015**.

7 Repository

Every team has access to its part of the Subversion repository of the course. You are free to use it for development, and we encourage you to frequently commit your changes in case your computer breaks down. The repository is set up by calling `setup.sh`, as described in the step-by-step guide. The assistants also check in bugfixes to the given code directly into the repository.

The URL of the repository is `https://svn.inf.ethz.ch/svn/meyer/ccc/trunk/teams/teamname`, where `teamname` stands for the name of your team.

8 Support

For help, consider using the course mailing list: se-ccc@lists.inf.ethz.ch. This list contains all students in the course in addition to the assistants.

There is also the possibility to ask questions during the office hours. Office hours are on Wednesday afternoon from 13:00 to 14:00 in RZ J6. Ask for Mischael Schill in RZ J3 if J6 is closed.

For resources on SCOOP and EiffelStudio, you can also consult the course website [1].

Any bugs in the provided code and the `ESTRING` library that get reported to us will be fixed and the patch applied directly to the repository.

9 Course website

Please check the course website [1] for potential updates of the project description and supporting material. The website also contains some hints and implementation details for the project that will be updated during the course.

References

[1] CCC. ETH Zürich. `http://se.inf.ethz.ch/courses/2015a_spring/ccc/`, 2015.

[2] VirtualBox. Oracle. `https://www.virtualbox.org/`, 2015.

[3] Bomberman. Wikipedia. `http://en.wikipedia.org/wiki/Bomberman`, 2015