consoden®
Winter Games

Virtual Machine Guide

Björn Weström <bjorn.westrom@consoden.se>

1 Introduction

This document provides setup instructions and some useful tips on the contents in the Virtual Machine for the team.

2 Performance tips

To get the best performance from the virtual machine, go to the virtual machine settings before you start it. Assign 4 CPUs to the VM (if possible). If you have plenty of free RAM, you can also increase the RAM assigned to the VM above 2 048 MB. Make sure that 128 MB video memory is assigned to the VM, and that 3D acceleration is checked.

3 Login details

Automatic logon is enabled, but if you need the login details to install extra software, here they are.
User name: cwg14
Password: cwg14

4 Choose language

Choose if you want to develop in C++, C# or Java. When reading the instructions below, the directory tank_player_xxx should be replaced with:

- tank_player_cpp for C++
- tank_player_cs for C#
- tank_player_java for Java

The source code example for your player is located in
`/home/cwg14/tank_game/tank_player_xxx/src`

5 IDE

It is fully possible to develop your player logic using a text editor and the build script described below. If you prefer to work in an IDE, we have setup Monodevelop for C++ and C#, and Eclipse for Java. If you use an IDE, you can run your player from within the IDE, you do not need to start it in a terminal. Monodevelop is started by typing

```
monodevelop
```

in a terminal. Eclipse is started by typing

```
eclipse
```

in a terminal.

6 Initial setup

6.1 Binary name

Default is `tank_player_cpp` (for C++, similar for the other languages), please change it to `tank_[team name]` as shown below. Since this will be the name of the binary, don't use any spaces and avoid special characters. If you didn't figure out a name yet, you can also just put your team number, for instane `tank_team5`.

Go to the directory `/home/cwg14/tank_game/tank_player_xxx/src` and edit the file CMakeLists.txt. Edit the string `tank_player_cpp` in this line:

```
set(proj_name tank_player_cpp)
```

6.2 Player name

Default is the same as the binary name, change it to your team name. No need to prefix it with tank_player, and it may contain spaces. This is the name that will be displayed in the game, and also the name that identifies your player in the Saifr SDK Core environment. Each process needs a unique identifier, so if you intend to run two instances of your player against each other, you need two binaries with different player names.

6.2.1 For C++

Go to the directory `/home/cwg14/tank_game/tank_player_cpp/src` and edit the file TankLogic.cpp. Edit the string `"tank_player_cpp"`:

```
const std::wstring TankLogic::PlayerName = L"tank_player_cpp";
```

(If you are unfamiliar with the L prefix of the quotes, this just means wide string. Don't remove it!)

6.2.2 For C#

Go to the directory `/home/cwg14/tank_game/tank_player_cs/src` and edit the file TankLogic.cs. Edit the string `"tank_player_csharp"`:

```
public const string PlayerName = "tank_player_csharp";
```

6.2.3 Java

Go to the directory `/home/cwg14/tank_game/tank_player_java/src` and edit the file TankLogic.java. Edit the string `"tank_player_java"`:

```
public final static String PLAYER_NAME = "tank_player_java";
```

7 Useful scripts

In your home directory (`/home/cwg14/`), there are a few useful scripts as described below.

7.1 startup.sh

This script initializes a terminal with multiple tabs, and starts up the necessary helper processes in separate tabs. The tabs are described below:

dose	Safir SDK Core main process. You should not need to watch this tab in the contest, unless you experience odd communication errors.
dope	Safir SDK Core helper process for data persistence. You should not need to watch this tab in the contest, unless you experience odd communication errors.
tank_engine	The game engine process. You should not need to watch this tab in the contest, unless your game is not starting.
tank_game_gui	A bash shell where you can start the tank_game_gui process.
Player1	A bash shell where you can start the player one process. We suggest that you use the included sample player, tank_sample.
Player2	A bash shell where you can start the player two process, your own player. See Tank Player below.
Build	A bash shell where you can issue build commands. See the build.sh script below.

dose, dope and tank_engine are all started automatically by the script, the other processes you must start manually.

Start the script by writing

```
./startup.sh
```

in a terminal.

7.2 build.sh

This script builds the code in `/home/cwg14/tank_game/tank_player_xxx/`. Start the script by writing

```
./build.sh
```

in a terminal. If you need to clean before the build, you can write

```
./build.sh --clean
```

in a terminal.

8 Tank Player

8.1 Sample

There is a sample player available for easier testing. You can start it by typing

```
tank_sample
```

in a terminal.

8.2 C++

Start your player process by typing your player process name

```
tank_player_cpp
```

in a terminal.

8.3 C#

Start your player process using mono by typing

```
tank_player_csharp.exe
```

in a terminal.

8.4 Java

Start your player process using java by typing

```
java -jar ~/safir/runtime/bin/tank_player_java.jar
```

in a terminal.

9 Tank game GUI

9.1 Performance

The tank_game_gui process has been optimized to make it usable also in a virtual machine (with no good graphics acceleration). However, when tank_game_gui is running, it may make your virutal machine slightly less responsive, which makes coding inconvenient. To avoid this, you can use some of the following methods:

- Shut down tank_game_gui when you don't need to watch it
- Slow down the refresh rate. You can enter the refresh rate (in Hz) as a command line parameter to tank_game_gui. For instance, `tank_game_gui 5` will start tank_game_gui with the reduced refresh rate of 5 Hz, which should work well for everyone. Warning: Don't set the refresh rate too high (above 50), it may make your virtual machine very unresponsive (and possibly make your whole computer very unresponsive).

9.2 Usage

To start a game, two player processes needs to be running. Start the game by choosing `New game...` in the menu. A dialog box will be displayed. If you just click `OK`, a new game will be started with a random game board. If you click `Browse...`, you can pick a game board file (a few game boards are included in your example code set). See below for creating you own game board files. You can also set the game pace. If you increase the game pace, it means that the animation of the game will be slower in the GUI, so it is easier to follow each move. Do note that the calculation time for each player is still 1 second, it is only the animation that is slower.

After the game is finished, you can restart the game by choosing `Restart game` in the menu. This will restart the game with the previous game board, but with starting positions switched between the players.

You can also choose to save the game board by choosing `Save game` in the menu. This can be useful if you made a random game board that you want to investigate further. The saved game board can then be loaded when you start a new game.

9.3 Display

During the game, the game board is displayed as a grid, with wall squares indicated as brick walls. The tanks and missiles are animated, and mines are indicated with mine images in the squares. On the left side and right side of the game board, the player names and joysticks are shown. The color of the player name (blue or red) is matched with the color of the respective tank. The joysticks show the current state of that players joystick. When the game is over, the game result is printed.

10 Game board files

Game board files are stored in `/home/cwg14/safir/runtime/tank_game/boards/`. The game boards are simple text files that you can edit in your editor. Each character represents one square on the game board, with one row of squares per line in the file. Make sure you use the same number of characters in each line. The following characters may be used in the game board files:

```
. Empty square
o Mine square
x Wall square
t Tank square - this is the starting position of a tank

Example - 5x5 game board with 4 wall squares:
..x..
.tx..
..x..
..xt.
.....
```

Each game board must include two tank squares, no more no less. Which tank square is used for player one depends on their order in the game board, but you can easily try both starting positions by using the Restart game function.

11 Troubleshooting

11.1 Issues

- The game does not start
- Error message when starting the player process

11.2 Solution

Restart the game environment. The easiest way is to close the whole terminal window created by the startup.sh script (closes all tabs). Then run the startup.sh script again. This kills any dead processes and restarts the game environment.