# Software Engineering

## User Guide

**Group 3**

**15th of May, 2015**

UNIVERSITÄT MANNHEIM

# Table of Contents

# Preface

The system was tested on Windows, Mac OS & Linux but was optimized for Windows & Mac OS.
The system was tested and optimized for at least the following hardware:

| | Windows | Mac OS |
|---|---|---|
| CPU | 1,9 GHz Intel Core i7 | 1,7 GHz Intel Core i7 |
| RAM | 4 GB 1600 MHz DDR3 | 8 GB 1600 MHz DDR3 |
| GPU | Nvidia GeForce GT 640M | Intel HD Graphics 5000 |
| Display Resolution | 1920 × 1080 | 1440 x 900 |

The program only runs correctly with the customized WeBots game world submitted as “worlds” folder.

# System Walkthrough

## Step 1:

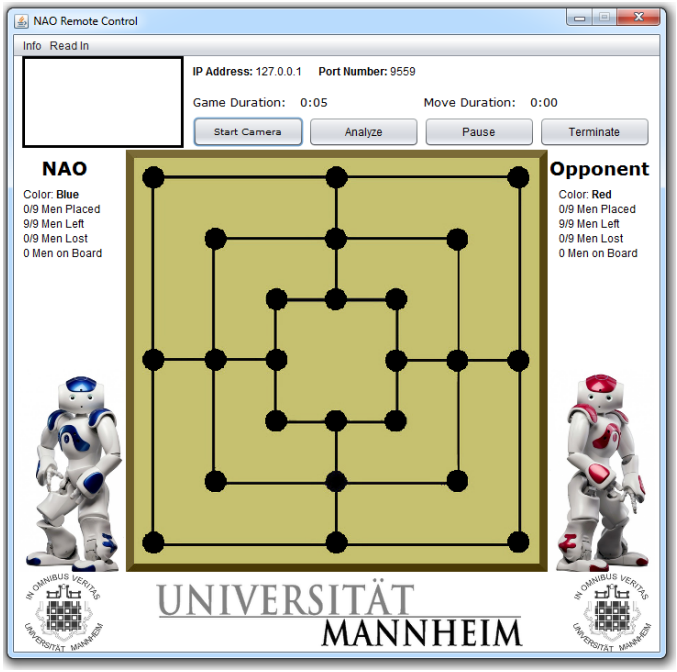
After launch of the .jar file, a „Connection Frame“ will open where the user has to enter a IP address and port number. To establish a connection to the entered IP address and port number the user has to press the „Connect“-Button.


## Step 2:

If the connection was established, the user has to choose a color. Otherwise, if the connection could not be established, the user will be informed about the problem.

## Step 3:

After choosing a color, the main frame will open. Now the user has different possibilities




### Step 3.1:

Start the camera stream of NAO through a click on the „Start Camera“-Button

### Step 3.2:

A click on the „Analyze“-Button causes NAO to start the first phase of his move: The analysis of the play board. After the analysis has finished, the user will be informed about it.

### Step 3.3:

After the play board was read in, the „Analyze“-Button will change to the „Next Move“-Button with which NAO starts phase two, its next move (see step 4).

### Step 3.4

There are also 2 menus about which the user can for example see the game rules, see the console output and the systems performance, make a visual play board read in.

## Step 4

The following describes the procedure NAO takes if he executes a move.

When the robot wants to take a gaming piece, it will signalize it with a grabbing motion and wait 10 seconds before going on with his move. Within these 10 seconds the user is able to give the robot the correct gaming piece, which it will detect with his lower camera.
After detection of the gaming piece, the user has 7 additional seconds to place the gaming piece into the robot’s hand. If the robot successfully grabbed the gaming piece, it will take it to the desired field and put it down, where the user should arrange the gaming piece if it is not standing correctly. Otherwise the user should place the gaming piece onto the field where the robot stops and kneels down. When the robot wants to place a gaming piece in the beginning of the game without taking another gaming piece, that has already been set, it will just go to the desired field and kneel down and stand up as signal for the user to place the gaming piece at the accordant field.
Same goes for removing a gaming piece of an opponent.

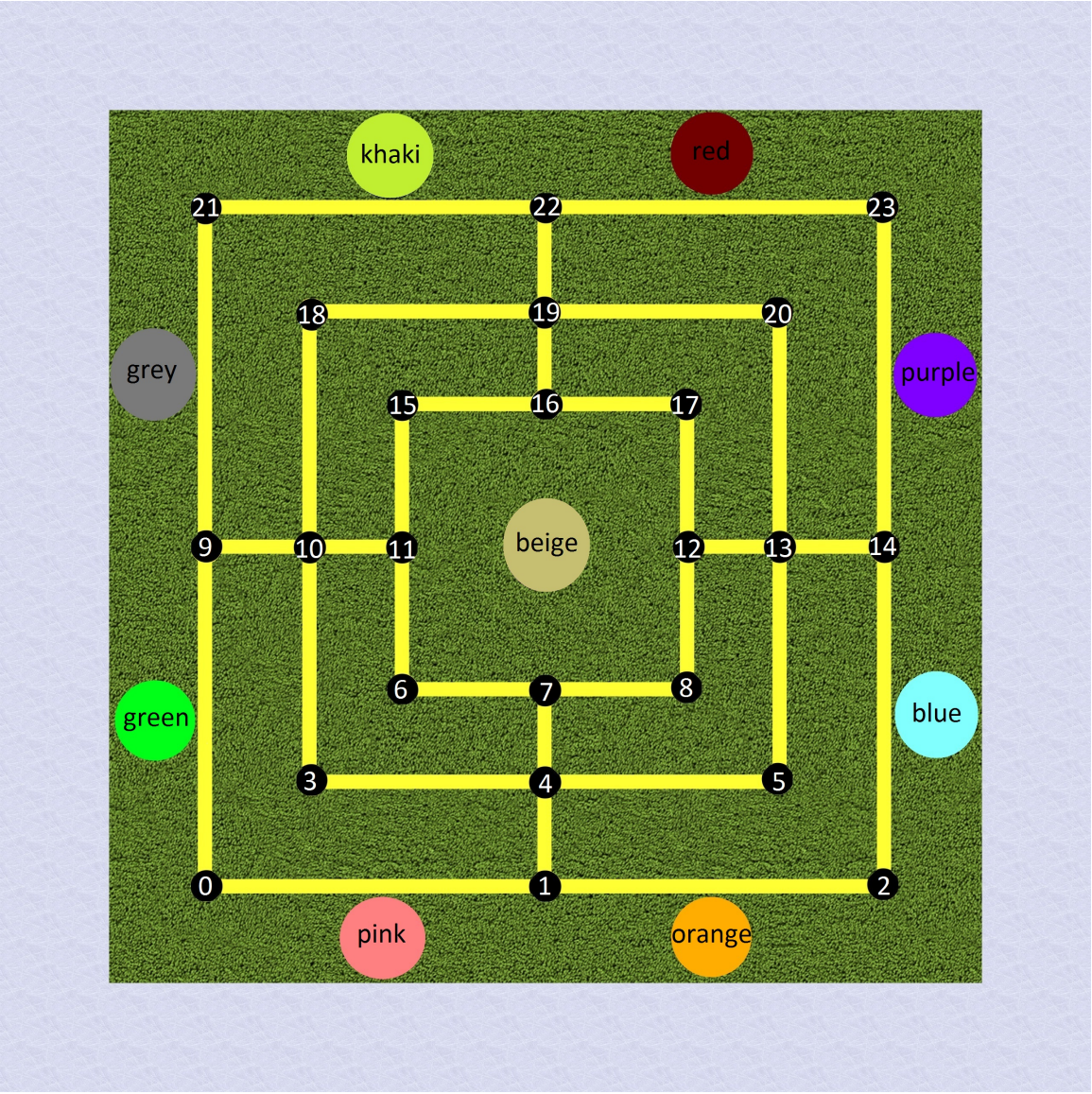
# Additional Information

## Assumptions

- The original gaming piece colors black & white are replaced with red & blue
- The color blue represents white, red represents black
- With the original colors white always starts and therefore with the new colors now blue starts

## Information about the customized game world

The following picture of the play board is helpful for checking, whether the robot is executing the moves correctly or not by comparing its actions with the console output.




## Existing Problems

- We have observered that after running the program and its simulation for a long duration, sometimes a java error can occur whose cause we believe is triggered by the simulation WeBots.
- As we created the .jar file we recognized that the binding of the Sigar API did not work correctly. Therefore some references of the system were not found and triggered an exception. The responsible part of the program is written as comments.

## Contact


- Nick Weber User Interface & Class Communication
- Julian Betz Artificial Intelligence & Data Representation
- Aljoscha Narr Events & Detection
- Wei Hao Lu Movement
- Jonas Thietke Logics & Data Representation
- Mike Siefert Exception Handling & Testing