Robobulls Operation Guide

This document details how to run the Robobulls software for running simulated or physical games, as well as resolutions for common issues with both the simulator and Yisibots.

This document is current with the following dependency versions:

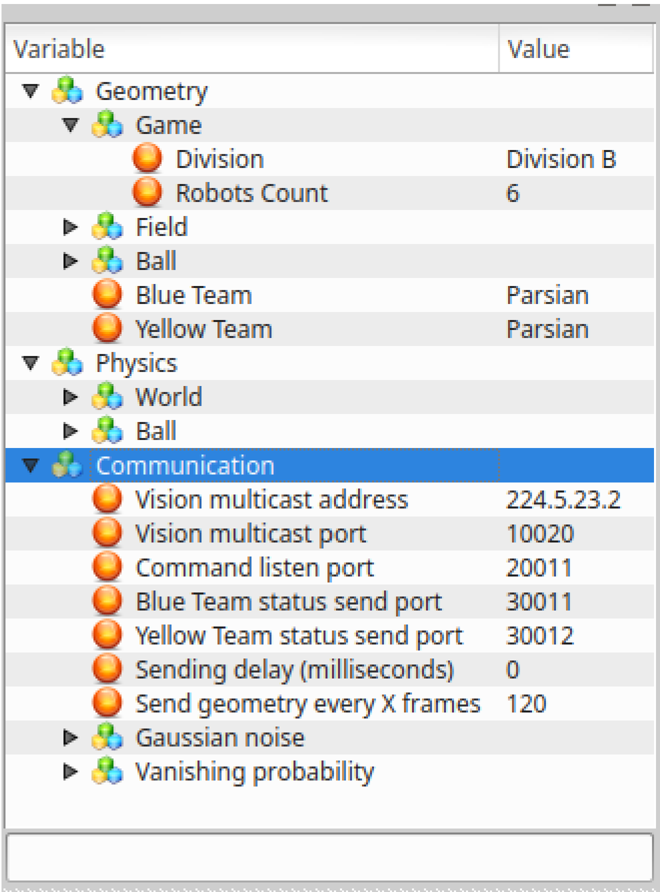
grSim	May 31, 2018	commit 45f0b3d
SSL-Refbox	Jun 18, 2018	commit a8498a4
SSL-Vision	Jun 8, 2018	commit f4bfeed

Running a Simulated 2-Team Game

There are 3 programs necessary to run a simulated game: grSim, SSL-Refbox, and the Robobulls dashboard.

To run a fully simulated game you must have **two** instances of the robobulls software installed, one of which must be in the team-blue-config branch, and the other in team-yellow-config branch (Note: This is an improper use of branches, and should to be changed to avoid the need for branch switching).

Make sure each instance of the software is installed in a different location. Both will need to be run, so that each team is represented. Before running Robobulls, change the simulated VISION_ADDRESS IP in communication.h to point at the IP reported in grSim's control panel (see image below).



Variable	Value
▼ Geometry	
▼ Game	
Division	Division B
Robots Count	6
▶ Field	
▶ Ball	
Blue Team	Parsian
Yellow Team	Parsian
▼ Physics	
▶ World	
▶ Ball	
▼ Communication	
Vision multicast address	224.5.23.2
Vision multicast port	10020
Command listen port	20011
Blue Team status send port	30011
Yellow Team status send port	30012
Sending delay (milliseconds)	0
Send geometry every X frames	120
▶ Gaussian noise	
▶ Vanishing probability	

Above: grSim control panel. Target the IP listed as Vision multicast address in Robobull's communication.h. Also note the division & robot count parameters.

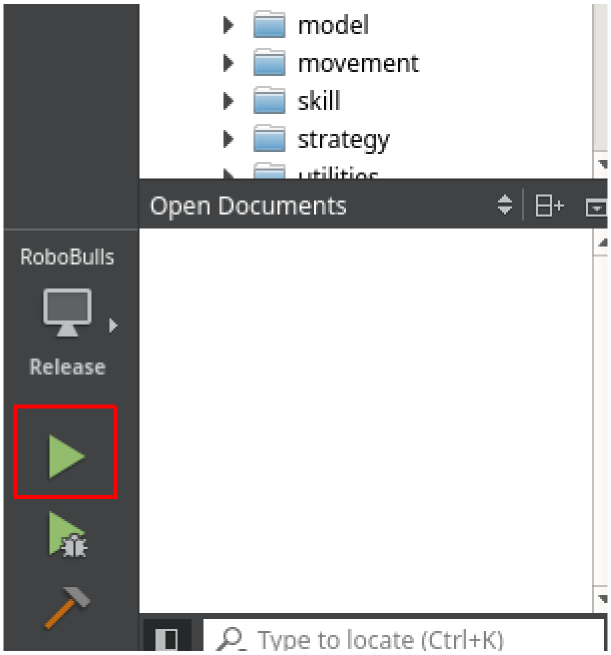
In the grSim folder, run the following command to start grSim's simulator:

```
./bin/grsim
```

In the SSL-Refbox folder, run the following command to start the referee program:

```
./sslrefbox
```

Next, open qtcreator, open the Robobulls project, and run the Robobulls dashboard.



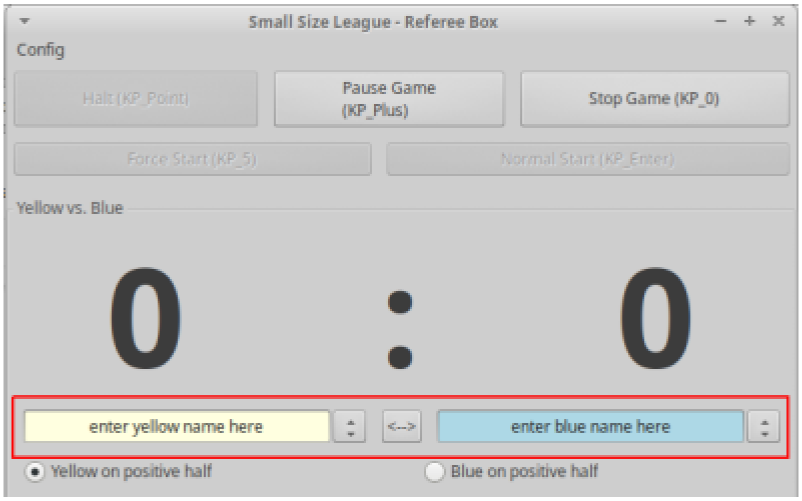
Above: QtCreator. Click the green arrow to compile & run the Robobulls dashboard.

Alternatively, run the following command in the team-blue-config Robobulls directory:

```
    ./Robobulls
```

Open a second Robobulls dashboard by repeating the steps above in the team-yellow-config directory.

In Refbox's program, and give a name to both teams then hit 'stop game'. Enter the goalie robots number for each team, then select a side to perform the kick off and hit 'start game'. The simulated game should then be viewable from either grSim or either of the robobulls dashboards GUI windows.



Above: SSL-Refbox. See the team name text boxes outlined in red. Note that the simulation will not function correctly unless both teams are named.

Running a Game with Robots

There are 3 programs necessary to run a game: ssl-refbox, robobulls dashboard, and ssl-vision. The details on preparing and running the SSL-Vision cameras is detailed in the SSL-Vision documentation.

In the SSL-Refbox folder, run the following command to start the referee program:

```
./sslrefbox
```

In the Robobulls folder, run the following command

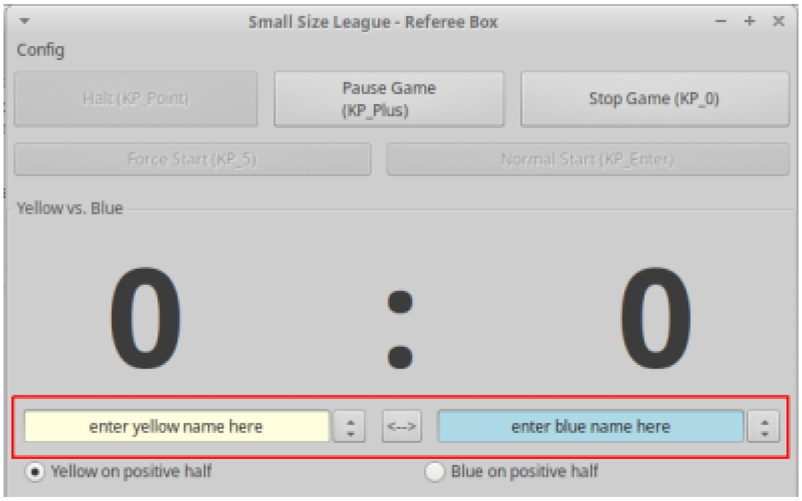
```
git checkout ssl-league-fixed
```

Run the following commands to compile, and run:

```
make
./Robobulls
```

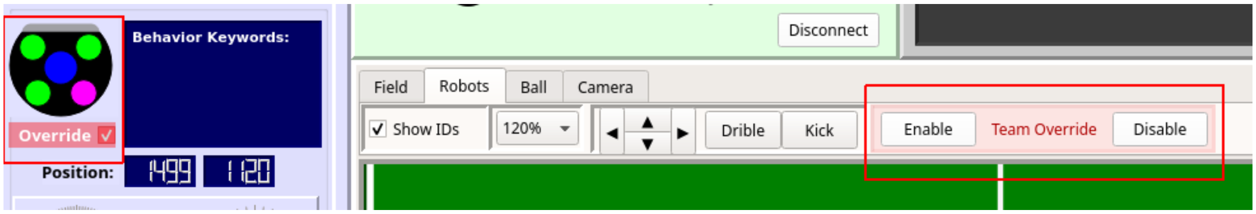
Alternatively, compile and run the project from QtCreator (see above image).

In Refbox's program, give a name to both teams then hit 'stop game'. Enter the goalie robots number for each team, then select a side to perform the kick off and hit 'start game'. The simulated game should then be viewable from either gr sim or either of the robobulls dashboards GUI windows.



Above: SSL-Refbox. See the team name text boxes outlined in red. Note that the simulation will not function correctly unless both teams are named.

If at anytime you need to halt the play of the game inside the robobulls dashboard, hit the stop game button inside refbox to bring the robots into a stand-by mode or halt to force stoppage. Alternatively, you can either hit the override checkbox of any robot to disable it. You can override the entire team by using the hotkey CTRL+o or the disable button along the top of the robobulls dashboard.

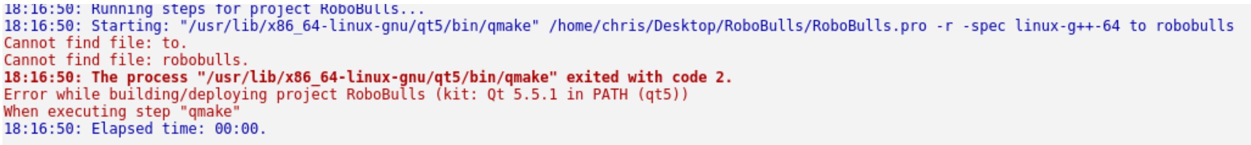


Above: Robobulls Dashboard. See the override checkbox (left) for individual bots, or the full-team override option (right). An overridden bot will not move autonomously.

Warning: If, for any reason, the camera fails to track a robot (insufficient color calibration in ssl-vision, robot exits camera field of view, etc), it will drive erratically, continually picking up speed until it is moving dangerously fast. The robot will need to be turned off manually by hitting the power switch located on the rear-left side of the robot's chassis, overridden in the Robobulls GUI, or issued a Halt command from the refbox. For this reason, we suggest lowering the robot's ACCEL_MULT in [include/config/move_parameters.h]() if you are not absolutely confident in the camera's calibration & responsiveness.

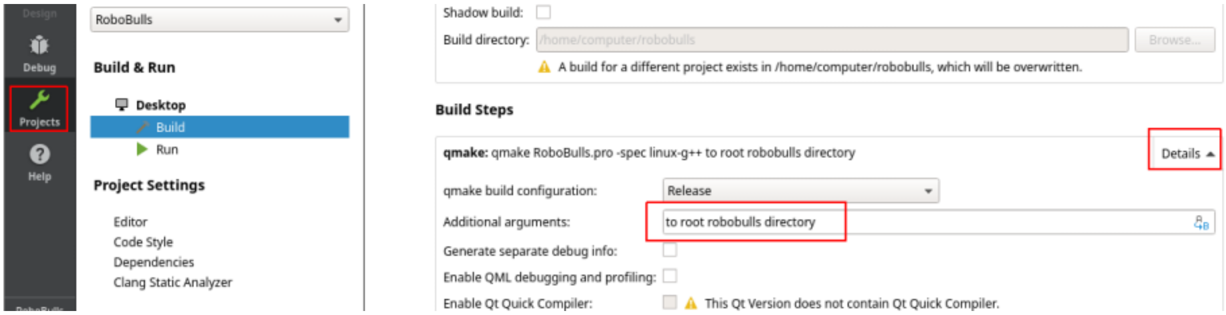
Known Issues & Workarounds

-Compilation issue when trying to compile from qtcreator



```
18:16:50: Running steps for project RoboBulls...
18:16:50: Starting: "/usr/lib/x86_64-linux-gnu/qt5/bin/qmake" /home/chris/Desktop/RoboBulls/RoboBulls.pro -r -spec linux-g++-64 to robobulls
Cannot find file: to.
Cannot find file: robobulls.
18:16:50: The process "/usr/lib/x86_64-linux-gnu/qt5/bin/qmake" exited with code 2.
Error while building/deploying project RoboBulls (kit: Qt 5.5.1 in PATH (qt5))
When executing step "qmake"
18:16:50: Elapsed time: 00:00.
```

If you receive the above error message while trying to compile from QtCreator, to resolve this issue open the projects tab on the left hand side of qt creator. Then, under the build steps tab hit the details button and delete anything in the additional arguments field. The project should then compile normally.



Shadow build: □

Build directory: /home/computer/robobulls

A build for a different project exists in /home/computer/robobulls, which will be overwritten.

qmake: qmake RoboBulls.pro -spec linux-g++ to root robobulls directory

qmake build configuration: Release

Additional arguments: to root robobulls directory

Generate separate debug info: □

Enable QML debugging and profiling: □

Enable Qt Quick Compiler: □ This Qt Version does not contain Qt Quick Compiler.

Above: QtCreator build settings. Open the menu with the projects button (left), expand the build details (right), and remove any invalid additional arguments (center)

-Connectivity problems when attempting to connect robots to the yisibot transmitter for non-simulated games.

Ensure that the serial port name in communication/yisirobcomm.cpp matches the USB device name used for the transmitter, found by looking in the file system's /dev/ folder under the name ttyUSB#, where # is a number assigned to the transmitter. Ex: "/dev/ttyUSB1"

Turn on the robot and allow it some time (if the robot's faceplace shows eyes, it is ready) to fully boot. Then, plug the yisi transmitter into the computer. Afterwards, run the robobulls dashboard. The yisi transmitter will blink left and right, then blink both lights simultaneously, then both lights will stop blinking and stay on, indicating successful connection to the bots (see unexpected behavior on connection issue below). At this point you should be able to manually move an overridden bot through the robobulls software by selecting it and using the WASD keys.

If this fails, repeat the process. The robots don't need to be powered off/on again. If the lights on the transmitter never transition from left/right blink to double blink on starting robobulls software, restart the machine. This may take many attempts.

-Robots frequently collide during pathing

Unresolved. Robobulls uses FPPA pathing (fast path planning algorithm), the details of which can be found here, and the implementation of which is found in movement/pathing/fppa_pathplanning files. Special note to commit 56bba26, where a 'personal bubble' was added around each bot, in which obstacles are ignored, as well as commit 08b6f67, which runs that algorithm as much as possible. Together, these changes seem to introduce behavior that results in bots immediately disregarding any obstacle that ever comes too close - further investigation required.

-Normalize speed function is implemented but not used

Currently this means robots increase in velocity indefinitely until they reach their target. Use of this function should set a hard cap for how fast they can move which is specified inside the robot soccer rule set (1.5 m/s).

Additional Resources

Robobulls Readme
Robobulls Installation Guide
Robobulls Operation Guide

Robobulls Development Goals
SSL-Robocup Ruleset
SSL-Refbox Documentation
SSL-Vision Official Documentation
SSL-Vision Robobulls Operation Guide
Yisibot Manual
Robobulls GUI Honors Thesis (Overview of Robobulls)

Also, see the Robobulls pre-2016 Doxygen HTML documentation, found by opening the Documentation.html file located in documents of the main project folder.

Contact Info

After reading all of the above documentation, if you need additional clarification on an issue detailed above and the current lab admins are not available or are unfamiliar with your problem, you can contact one of the previous team members for more information.

May-Aug 2018 Team Members:
nherbert2@mail.usf.edu

- Familiar with Yisirobot movement code, out-of-lab installation. Authored readme, devgoals, installation guide, operation guide. (8/03/18)

kellerc@mail.usf.edu