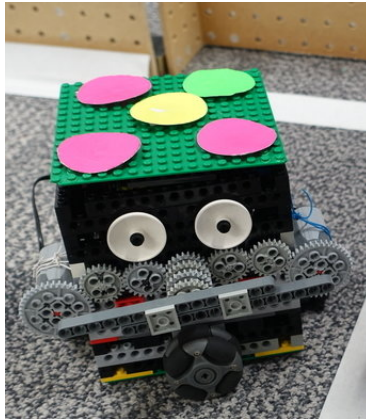


# System Design Project User Guide

Group 2A

## 1 Robot

### 1.1 An Overview of F.R.E.D



The robot F.R.E.D has been manufactured for playing two-a-side football, similar to the Robocup competition small sized league. This football robot has commands which will enable you to begin playing with a ball, such as kicking a ball given a specified distance. F.R.E.D is a small size of roughly 20cm x 18cm x 18cm to enable a football pitch of the mere size 300cm x 220cm.

### 1.2 What to Install

All necessary source files and libraries can be found at:

[https://bitbucket.org/sdp\\_group\\_2/firmware](https://bitbucket.org/sdp_group_2/firmware)

The main source file (Firmware2.ino) needs to be uploaded into the Arduino Uno Board. The source file includes all the functions needed to control the movement. It also sets up the ports from the Arduino Uno Board. In order to upload it you need to have installed Arduino IDE on your work station. To be able to compile the main source file, you need to add the SDPArduino and SerialCommands libraries. (Sketch → Import Library → Add Library → select zipped library).

Insert the *Ciseco SRF Stick* into one of the USB ports on your work station. The *Ciseco SRF Stick* and the Arduino board should connect automatically. If they do not connect, reset both of them and reconfigure them.

## 1.3 Technical Review of Commands

### 1.3.1 Moving

F.R.E.D uses four motors for moving. A motor with a wheel attached, is fixed to each of the four sides of the robot. These bi-directional wheels allow for holonomic movement, meaning F.R.E.D can move in any direction whilst facing a desired direction (eg. facing the ball).

### 1.3.2 Kicking

During a game of two-a-side football, each robot has to be able to act in a defensive role by moving the ball from the defence area, by grabbing and kicking for example, or in our case, kicking. Our version of F.R.E.D uses one large propeller for kicking. This powerful kicker is located on the front-facing side, and is controlled by a motor at the left and right sides of the robot. The motors will spin in a particular direction, which is determined by the position of F.R.E.D to the ball, controlled by the Strategy System. The direction of spinning will change such that it will always kick towards the opponents half. This increases the chance of a goal against the opponent and decreases the chance of an own goal.

## 1.4 How to Use

### 1.4.1 Getting Power

1. Connect the *Ciseco SRF Stick* into a USB port in your PC.
2. Remove the top 'lid' of F.R.E.D. Attach the battery connector (red and black wires) to the battery pack containing 4 Lithium rechargeable batteries. Make sure batteries have voltage between 12V and 16V.
3. Type `arduino` into a terminal, which will bring up another window. If this is unresponsive, refer to Section 1.5.1.
4. Click on the `Serial Monitor` icon, which will bring up another window. If this is unresponsive, refer to Section 1.5.2.
5. Choose from the commands below and watch your robot move!

### 1.4.2 Movement Commands

Command to Enter	Translation
m <i>[front motor power] [back motor power] [left motor power] [right motor power]</i>	Robot moves at speed/direction according to motor powers given. (Powers must be integers ranging from -255 to 255)
f	Force stop moving (only stops wheels).
h	Halt - Stop everything (good emergency mechanism).
ping	Returns "pang". Checker for working connection.

### 1.4.3 Kicking Commands

These are the commands which were specific for our version of Fred (the propeller). You may use these or you may choose to implement your own.

Command to Enter	Translation
kick -1	Propeller moves in anti-clockwise direction.
kick 1	Propeller moves in clockwise direction.
kick 0	Propeller stops.

## 1.5 Troubleshooting

### 1.5.1 Terminal

After following instructions on how to upload Firmware2.ino as specified in section 1.2, if typing `arduino` into a terminal proves unresponsive please try the following:

- Exit terminal window and do Step 3 from Section 1.4.1 again.
- Disconnect and reconnect the *Ciseco SRF Stick* from the USB port.

### 1.5.2 Serial Monitor

If Serial Monitor window does not appear when you click on the **Serial Monitor** icon, please try the following:

- Disconnect and reconnect the *Ciseco SRF Stick* from the USB port.
- Connect the *Ciseco SRF Stick* into an alternative USB port.
- Restart the Arduino IDE.
- Contact *The A team*.

This problem is caused by the Arduino IDE binding itself to a single port regardless of what you are doing (reproducible by using the USB cable and then switching to SRF stick without restarting the Arduino IDE).

### 1.5.3 Unresponsive Robot

If F.R.E.D does not respond to commands sent on the **Serial Monitor**, please try the following:

- Disconnect the battery connector from the battery pack and reconnect.
- Check battery charge is at least 12V. If it is not, replace batteries or charge existing batteries.
- Contact *The A team*.

## 2 Communications

Once the Strategy module is turned on, it automatically creates a communications class and tries to find F.R.E.D. Progress on this can be seen in the *SDPConsole* window, once it says "*Found robot! Yay!*", the robot has been found. While it is looking for the robot, you may unplug the SRF stick, plug it in again, or restart F.R.E.D to get it to connect, as doing this will not break anything.

As long as you ensure that Fred responds with "pang" when he receives "ping", this module will work.

## 3 Vision System

### 3.1 Overview of the Vision System

The vision system was designed in such a way that makes sure almost no code alterations need to be done when reusing it for a different project. It comes with a graphic user interface that allows changing virtually every setting of the robot detection. It can be run straight out of the box and will work to some extent in almost any circumstances. *The A team* recognizes the hardship that comes with reusing code, so the Vision System will provide informative messages to help the user resolve any problems that they may encounter. That is, the code will *run*. It may not do exactly what you want it to do, but once run, it will guide you through the debugging process.

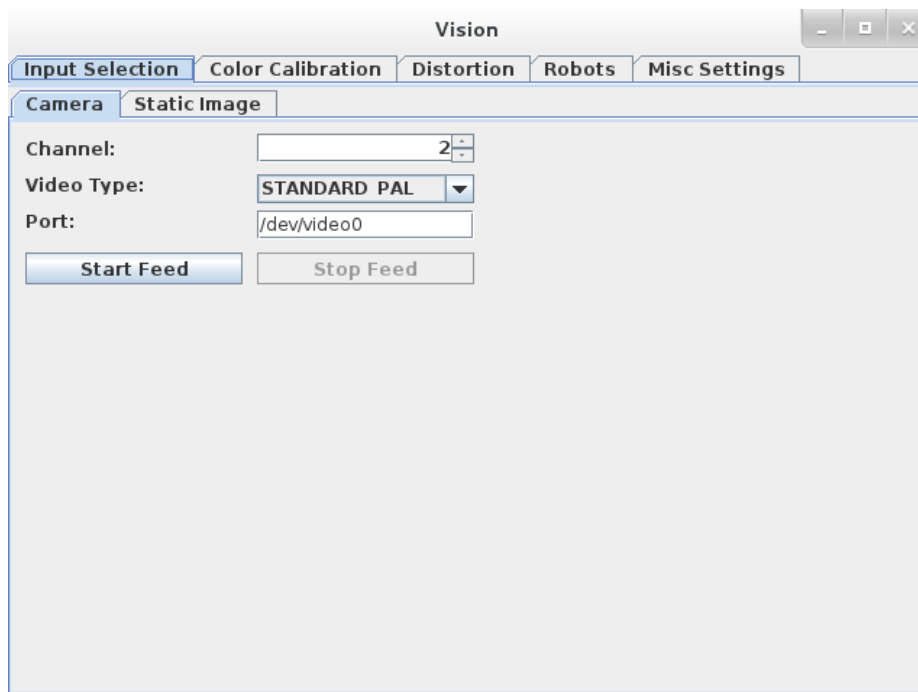
## 3.2 Running the code

To run the code, clone the repository [https://bitbucket.org/sdp\\_group\\_2/vision](https://bitbucket.org/sdp_group_2/vision) and follow the relevant instructions:

1. Download IntelliJ.
2. Make yourself an IntelliJ student account (JetBrains lets students use their software for free).
3. Go to <http://fred.rovder.com/cloning.html> for a video tutorial, showing how to set up IntelliJ. (This process is rather complicated, so follow the video carefully)

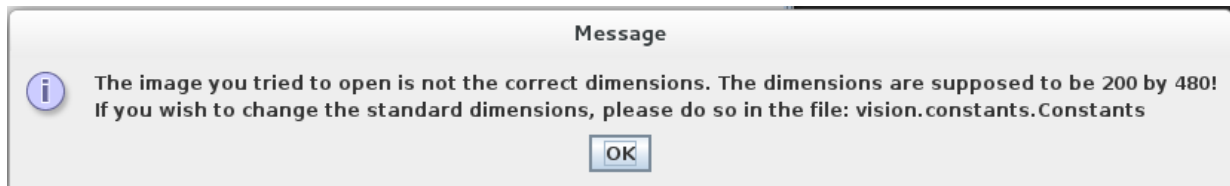
## 3.3 Necessary Code Alterations

The Vision System will work straight out of the box. Even if any internal settings are not appropriate for your needs. Once run, you will be presented with the following window:



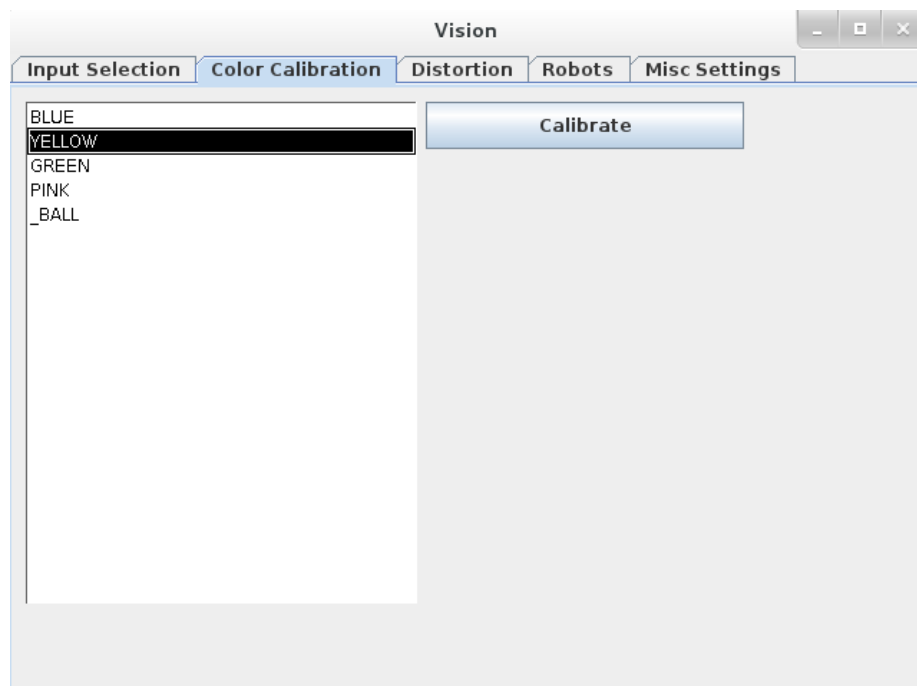
### 3.3.1 Widths and Heights

For optimization purposes, there are four constants that need to be rewritten in the code. They are located in the `vision.constants.Constants` class. These are the input image dimensions (in pixels) and the physical pitch dimensions (in centimetres). If you fail to set up the pitch dimensions, the robot analysis may be skewed. If you fail to rewrite the input dimensions, running the camera feed will produce this error message:



### 3.3.2 Colours

The colours you want the Vision System to detect are specified in the enum `vision.colorAnalysis.SDPColor`. Once the enum is changed, the GUI changes to accommodate the change. So if you wish to have more colours in the system than our team needed for our purposes, you can easily add them into that enum. Once added, they will appear in the Colour Calibration tab:



### 3.3.3 Robots

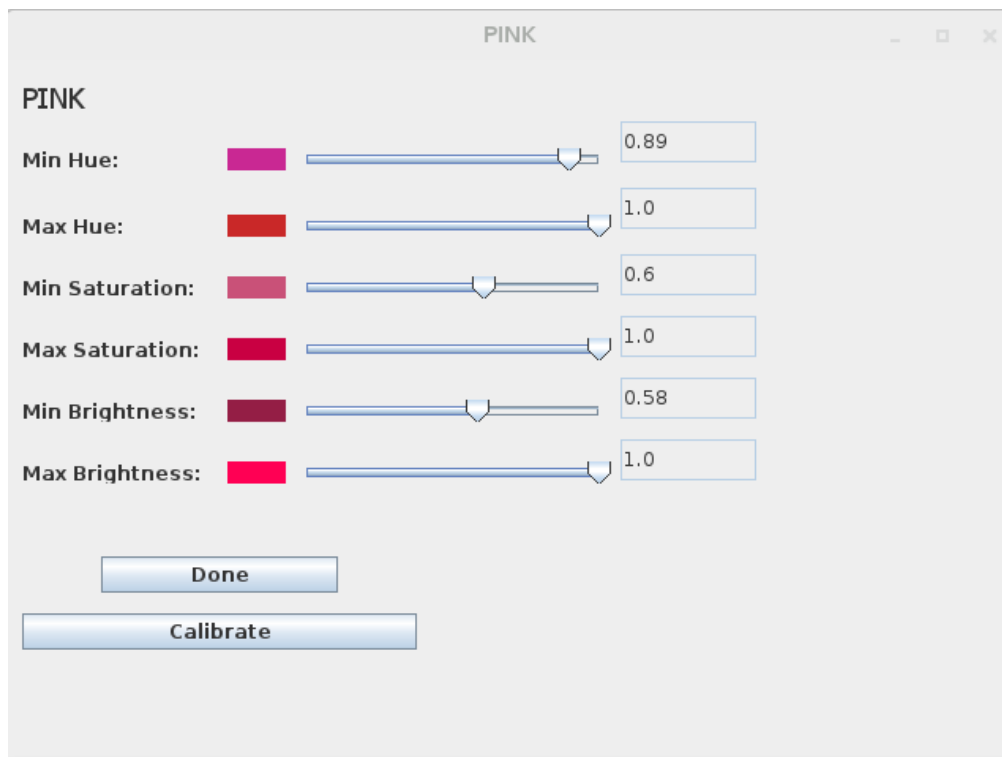
The robots you want the Vision System to detect are specified in the enum `vision.RobotType`. If you have a different configuration of robot plates, this is the only section of the code you may need to change. This is because the robot detection was optimized for our robot plates, which closely resembled the RoboCup robot plates.

## 3.4 The User Interface

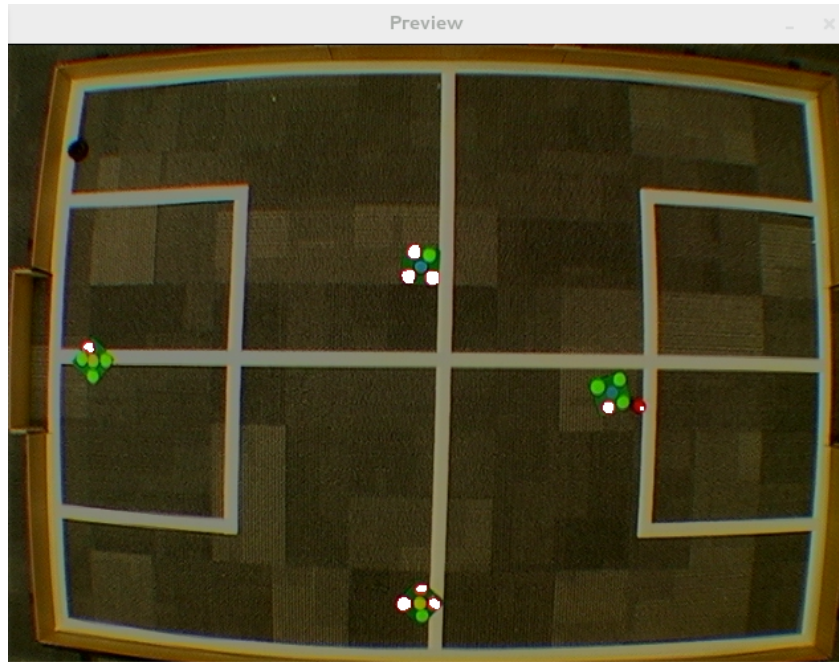
When you run the code, the interface allows you to calibrate colors, adjust the distortion and choose from a variety of input methods. They are further clarified below:

### 3.4.1 Colour Calibration

All the colours can be calibrated in the Colour Calibration tab. Simply select a colour from the colours tab and click *Calibrate*. The following window will come up:

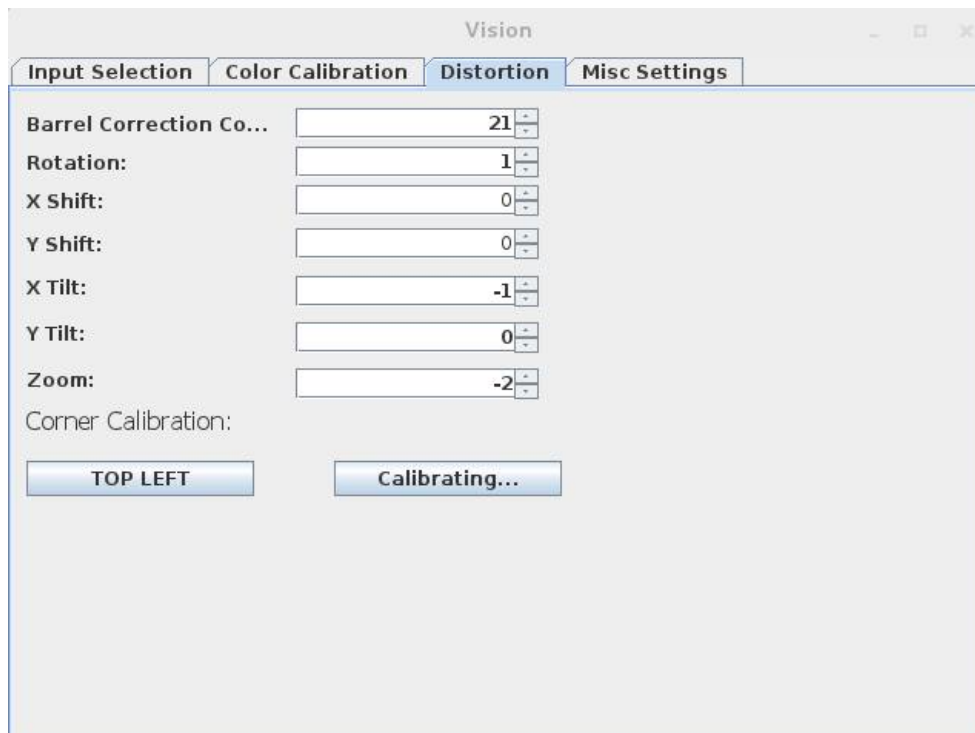


You may then pull the sliders around to change the ranges and see the effect on the *Preview* window. Clicking the *Calibrate* button toggles colour picking mode, when in this mode, clicking on the preview window will save the colour of the pixel you clicked on as the colour you are calibrating.



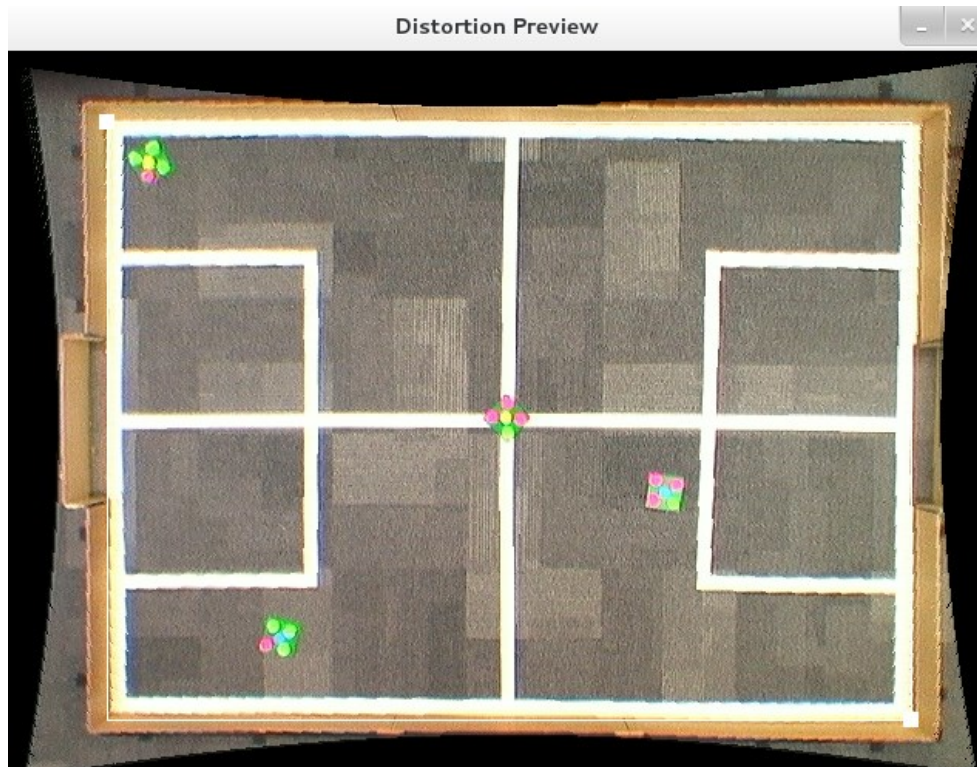
### 3.4.2 Distortion Setup

The Distortion tab in the Vision control window allows you to dynamically calibrate distortion on the fly:



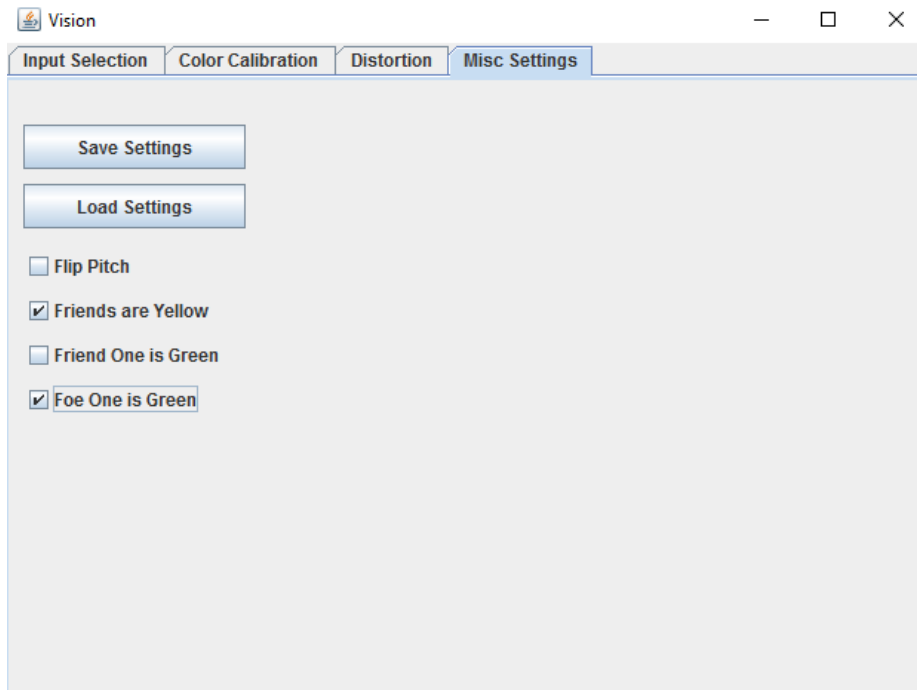


Even though the Vision System does not undistort the frames themselves, it will undistort an entire frame once with every change in the distortion settings (so you can see what the undistortion does). You can see this preview in the Distortion Preview window:



### 3.4.3 Configuring the current game

In order to correctly detect the robots and pitch, four checkboxes have been added to the Misc Settings tab. Toggling these checkboxes changes the robot settings and the new settings are instantly reflected in the `DynamicWorlds` passed onto the strategy module.



### 3.5 Connecting the Vision System to your code

To run the Vision System with your code, follow these instructions:

1. One or more of your classes should implement the `VisionListener` interface. These classes are then able to receive output from the Vision System.
2. Instantiate the `Vision` class. Located in `vision.Vision`. Instantiate it with 

```
Vision vision = new Vision();
```
3. Add your `VisionListener` classes as listeners: 

```
vision.addVisionListener(yourClassHere).
```

### 3.6 Running

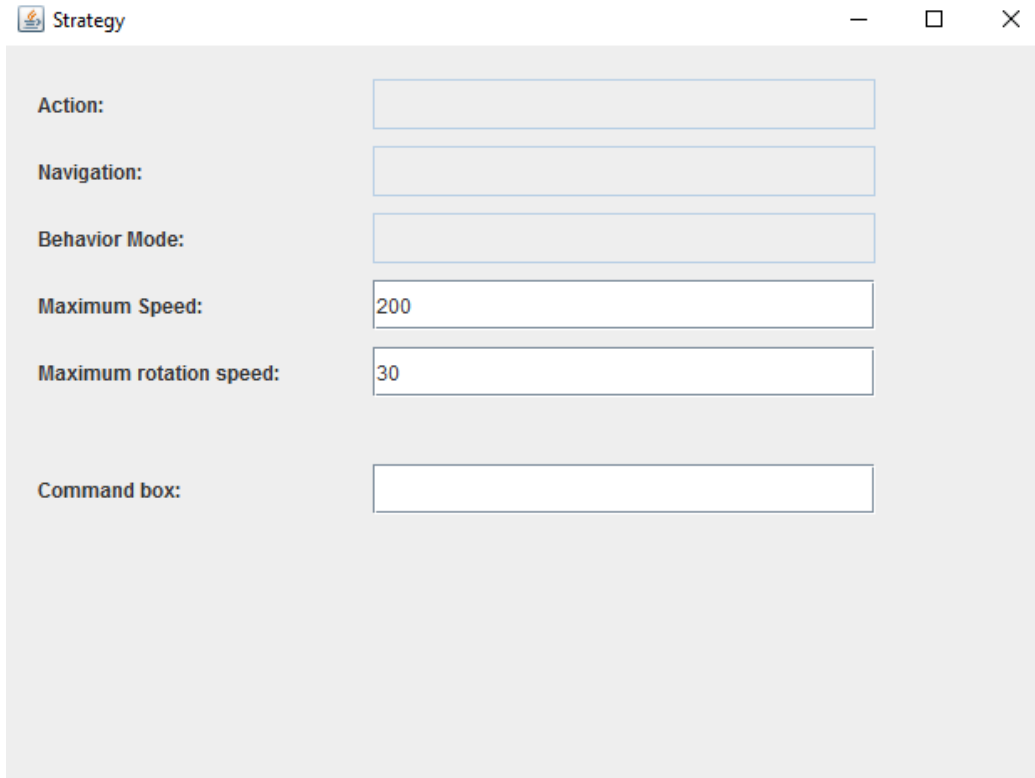
In order to save time when running the vision system, the following command like arguments allow you to do some things on start-up:

1. `--load [path]` - Loads the saved vision system settings.
2. `--videoChannel [channel]` - Sets the camera feed channel to the provided number.
3. `-stream` - Starts the camera feed.

This is very helpful when debugging the Strategy Module.

## 4 Strategy

Using the Strategy module is very simple and is done using the Strategy window:



The screenshot shows a window titled "Strategy" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains several input fields:

- Action:** An empty text input field.
- Navigation:** An empty text input field.
- Behavior Mode:** An empty text input field.
- Maximum Speed:** A text input field containing the value "200".
- Maximum rotation speed:** A text input field containing the value "30".
- Command box:** An empty text input field.

When running, this window will inform you which Action is currently being executed, which Behaviour Mode is being used (when playing) and the Navigation System currently in the pipeline (A\* or Potential Fields).

The Maximum Speed and Maximum rotation speed sections allow you to change Fred's speed while playing a game (Maximum of 255 for each, although the default settings are usually the best). You may want to change these values when the batteries are too powerful (Fred has no problem moving very fast and you are more likely to want to *slow* him down than to speed him up).

The Command box is the main control interface. Positioning your cursor onto the command box allows you to change the currently executed action just by hitting the correct key. These are the keys implemented:

<b>Key</b>	<b>Action</b>	<b>Description</b>
d	DefendGoal	Defend the goal.
k	OffensiveKick	Kick the ball.
s	GoToSafeLocation	Go to our goal while avoiding the ball.
b	Behave	Play Football.
<i>Space</i>	Waiting	Halt.
<i>Numbers</i> (on the numpad)	Goto	Makes Fred go to the corresponding location on the pitch. If the numpad was the pitch, '4' is the friendly goal, '6' is the enemy goal, '5' is the center of the pitch, '7' is one of the corners etc.