**RoboWars Use Case Specification**

**SYSC 4907  
October 1st, 2010  
Project #34  
  
Alexander Craig - 100741774  
Alexander Dinardo - 100737587  
Steven Legere - 100735141  
Mike Wright - 100741876**

# 1 – Actor Definitions

**Server Administrator:** The server administrator actor represents a user who is physically present at the location of the robot arena and server. This user has access to a configuration GUI running on the server machine, and is primarily responsible for configuring the camera to be used with the system and ensuring that the robots are correctly positioned prior to mobile users initiating gameplay. The server administrator can also handle any physical problems with the robots, and may force a reset of robot positions if necessary.

**Robot:** The robot actor represents any robot which will be registered for remote control with the system. Although the system is specifically designed for use with LEGO Mindstorm NXT robotics kits, any robot which supports Bluetooth communication and the system’s custom communication protocol can be used.

**Mobile User:** The mobile user actor represents any user of the system who connects remotely from a mobile platform. Note that the module of code running on the mobile user’s smartphone is considered within the scope of the robotics control system for the purposes of these design documents. All mobile users have the ability to connect to the system lobby and chat with other users.

**Player:** A player is the default state of all mobile users when they connect to the system. This actor represents a mobile user who intends to make full use of the robotics control system. They are included in the rotation of users who are paired to the registered robots, and they have the ability to modify game options and launch games.

**Spectator:** A spectator represents a mobile user who has specifically opted-out of robotics control. Spectators can still view gameplay in progress and chat with other users, but they will not be considered in the robot control rotation.

# 2 – Use Cases

## 2.1 – Server Administration Use Cases

|  |  |
| --- | --- |
| **Use Case Name** | **Configure wired camera** |
| **Description** | The server administrator selects the camera to use from a list of detected cameras, and configures the position, facing, and field of view of the selected camera. |
| **Precondition** | The server software is running. |
| **Primary Actor** | Server Administrator |
| **Secondary Actors** | None |
| **Dependencies** | INCLUDE: **Display local video feed** |
| **Basic Flow** | 1. The server administrator selects a “configure camera” option from the server administration GUI. 2. The server administrator selects the desired camera from a list of cameras detected by the system. 3. The server administrator sets the co-ordinates and heading of the camera (in relation to the robot’s arena). 4. The server administrator specifies the field of view of the selected camera (necessary for augmented video rendering calculations). 5. The system displays the camera’s video feed through the server administration GUI (INCLUDE USE CASE **Display local video feed**). |
| **Post Condition** | A valid camera is configured for use with all further activities requiring a video feed. |

|  |  |
| --- | --- |
| **Use Case Name** | **Display local video feed** |
| **Description** | A live video feed of the “arena’ containing the remote robots is read from the attached camera and displayed through the server administration GUI. |
| **Precondition** | The server software is running.  A valid camera must have been previously configured for use with the system. |
| **Primary Actor** | Server Administrator |
| **Secondary Actors** | None |
| **Dependencies** | None |
| **Basic Flow** | 1. The server administrator selects a “view video feed” option from the system’s server administration GUI. 2. A video stream is read from the system’s wired camera. 3. The video stream is displayed in a separate frame of the system’s administration GUI. |
| **Post Condition** | The video stream is visible through the server administration GUI. |

|  |  |
| --- | --- |
| **Use Case Name** | **Reset robot positions** |
| **Description** | The server administrator forces a reset of the robot positions in the system’s internal state. This should be performed if the system’s positioning information for a robot becomes out of sync with the robot’s actual position for any reason. |
| **Precondition** | The server software is running.  A game is not in progress. |
| **Primary Actor** | Server Administrator |
| **Secondary Actors** | None |
| **Dependencies** | None |
| **Basic Flow** | 1. The server administrator selects a “reset robot positions” option from the server administration GUI. 2. The position of all robots currently registered with the system is reset to their default starting positions (this should be accompanied by the administrator physically relocating the robots to their starting positions). |
| **Post Condition** | The system records all registered robots as being positioned at their default starting position. |

## 2.2 – Robot Interaction Use Cases

|  |  |
| --- | --- |
| **Use Case Name** | **Register robot for remote control** |
| **Description** | A remote robot connects and registers with the system. |
| **Precondition** | The server software is running. |
| **Primary Actor** | Robot |
| **Secondary Actors** | None |
| **Dependencies** | None |
| **Basic Flow** | 1. The robot initiates a Bluetooth connection with the server. 2. The robot sends a string indicating which version of the command and positioning protocol it supports. 3. The system responds with an acknowledgement. 4. The robot is added to the system’s list of active remote robots. |
| **Post Condition** | The robot will be considering for pairing to mobile users after registration has completed. |

|  |  |
| --- | --- |
| **Use Case Name** | **Unregister robot for remote control** |
| **Description** | A remote robot unregisters from remote control and disconnects from the system. |
| **Precondition** | The robot has been previously registered for remote control. |
| **Primary Actor** | Robot |
| **Secondary Actors** | None |
| **Dependencies** | INCLUDE: **End game** |
| **Basic Flow** | 1. The robot sends an “END” signal to the system. 2. The system responds with an acknowledgment. 3. The Bluetooth connection is torn down. 4. Any game in progress using the robot is ended (INCLUDE USE CASE **End game**). |
| **Post Condition** | The system does not consider the robot for any further pairing with remote users. The remote may be disabled once it has unregistered from the system. |

|  |  |
| --- | --- |
| **Use Case Name** | **Update robot position** |
| **Description** | A robot asynchronously updates the system to reflect its current position. |
| **Precondition** | The robot has been previously registered for remote control. A game is in progress. |
| **Primary Actor** | Robot |
| **Secondary Actors** | None |
| **Dependencies** | None |
| **Basic Flow** | 1. Any movement of the robot’s motors triggers a position update. 2. The robot calculates its current position based on a known last position and calculated change in position (depends on movement mechanism). 3. The robot’s current position is transmitted to the system through the Bluetooth connection. 4. The system’s internal state is updated to match the robot’s reported position. |
| **Post Condition** | The robot position stored by the system is synchronized with the position locally calculated by the robot. |

|  |  |
| --- | --- |
| **Use Case Name** | **Join waiting lobby** |
| **Description** | A mobile client initializes a connection to the server. |
| **Precondition** | The server software is running. |
| **Primary Actor** | Mobile user |
| **Secondary Actors** | None |
| **Dependencies** | None |
| **Basic Flow** | 1. The mobile user initiates a connection to the system through standard internet protocol. 2. The system responds with an acknowledgment to ensure the connection has been established. 3. The mobile user provides a user name to identify them in the current session. 4. The mobile user is added to end of the rotating list of active users. 5. The mobile user is presented with the server lobby (a list of connected user names and a chat window). |
| **Post Condition** | The mobile user is identified by a unique user name and is added to the server lobby. |

## 2.3 – Server State and User Management Use Cases

|  |  |
| --- | --- |
| **Use Case Name** | **Send chat message** |
| **Description** | A mobile user sends a chat message to be displayed to all other connected mobile users. |
| **Precondition** | The mobile user has established a connection to the server.  The mobile user is either in the game lobby or in a running game. |
| **Primary Actor** | Mobile user |
| **Secondary Actors** | None |
| **Dependencies** | None |
| **Basic Flow** | 1. The user enters a string of text to be displayed to other mobile users. 2. The system displays the text message to all connected users along with the user name of the originating mobile user. |
| **Post Condition** | The chat message is displayed on the mobile client of all connected mobile users. |

|  |  |
| --- | --- |
| **Use Case Name** | **Opt-out of robot control** |
| **Description** | A mobile user who wishes to view rather than control the remote robots may choose to opt-out of robot control and become a pure spectator. |
| **Precondition** | The mobile is user is in the game lobby.  A game is not currently in progress. |
| **Primary Actor** | Spectator |
| **Secondary Actors** | None |
| **Dependencies** | None |
| **Basic Flow** | 1. The user selects an option (checkbox) to opt-out of robot control. 2. The user is identified in the waiting lobby with a “(Spectator)” tag next to their user name. 3. The user is moved to the bottom of the lobby waiting list. 4. When a game is launched, the user is not considered for pairing to a remote robot. |
| **Post Condition** | A spectator will never be placed in control of one of the remote robots. |

|  |  |
| --- | --- |
| **Use Case Name** | **Select game type** |
| **Description** | A player selects the type of game to play prior to starting a new game. |
| **Precondition** | The player is in the game lobby.  The player has not opted-out of robot control (not a spectator).  A game is not currently in progress. |
| **Primary Actor** | Player |
| **Secondary Actors** | None |
| **Dependencies** | None |
| **Basic Flow** | 1. The player is presented with a selection (drop down box) of all supported game modes (“Tank Simulation” and “Light Cycles”). 2. The player selects one of the provided game modes. 3. The selected game mode is displayed in the waiting lobby to all connected users. |
| **Post Condition** | When a game is launched, the selected game mode will be used unless another selection is made before the game begins. |

|  |  |
| --- | --- |
| **Use Case Name** | **Pair player to robot for remote control** |
| **Description** | A specific player is paired to a specific robot for the duration of a single game. Any commands from the player will control only the robot paired to the specific player. |
| **Precondition** | A currently unused robot and unpaired player must be connected to the system.  A game is not in progress. |
| **Primary Actor** | Player |
| **Secondary Actors** | Robot |
| **Dependencies** | None |
| **Basic Flow** | 1. The system selects the first player in the waiting lobby to be paired to a robot. 2. The system selects the first available robot to be paired to the player. 3. The system flags the robot as “in use” (only one player can control a robot) 4. The system moves the paired player to the bottom of the waiting list of players (provides rotation of robot use when more players than robots are connected). |
| **Post Condition** | All further commands from the paired player are propagated only to the paired robot. |

## 2.4 – Virtual Gameplay Use Cases

|  |  |
| --- | --- |
| **Use Case Name** | **Launch game** |
| **Description** | A player in the lobby initiates a game. The lobby is replaced with an augmented video feed, and players are paired to robots for remote control. Virtual gameplay begins. |
| **Precondition** | The server software is running.  A game must not be in progress.  At least two players must be in the server lobby.  At least two robots must be registered with the system.  A game type has been selected.  A valid camera must have been previously configured for use with the system. |
| **Primary Actor** | Player |
| **Secondary Actors** | Spectator, Robot |
| **Dependencies** | INCLUDE: **Pair player to robot for remote control**  INCLUDE: **Display augmented video** |
| **Basic Flow** | 1. A player selects a “begin game” option from the system’s server lobby. 2. Two or more players are paired to robots for remote control depending on the number required by the game type (INCLUDE USE CASE **Pair player to robot for remote control**). 3. An augmented video feed is displayed to all connected mobile users (INCLUDE USE CASE **Display augmented video**). 4. The system changes state to reflect that a game is in progress. 5. The system starts real time virtual simulation of gameplay and concurrently activates remote control of robots by remote users. |
| **Post Condition** | At least two players are paired to robots.  Robot remote control is activated.  Virtual gameplay simulation is in progress. |

|  |  |
| --- | --- |
| **Use Case Name** | **Display augmented video** |
| **Description** | The live video feed of the camera overlooking the robot arena is combined with the internal game state representation to produce virtual elements overlaid on the live video feed. |
| **Precondition** | The server software is running.  A valid camera must have been previously configured for use with the system. |
| **Primary Actor** | None |
| **Secondary Actors** | Mobile User |
| **Dependencies** | None |
| **Basic Flow** | 1. The system selects a video feed based on the previously configured camera settings. 2. The system renders virtual elements represented within the game state (such as name tags for the robots, any virtual obstacles in play, virtual projectiles, and UI elements such as health bars). 3. The virtually rendered elements are overlaid on the live video feed. 4. The combination of live video and virtually rendered elements are displayed to all connected mobile clients. |
| **Post Condition** | The augmented video feed is visible to all connected mobile clients. |

|  |  |
| --- | --- |
| **Use Case Name** | **Move robot** |
| **Description** | A mobile player issues a command for a robot to move, which is passed to the robot by the system. |
| **Precondition** | The player is currently paired to a robot.  A game is in progress. |
| **Primary Actor** | Player |
| **Secondary Actors** | Robot |
| **Dependencies** | None |
| **Basic Flow** | 1. The player issues a command for a robot to move (either through tilt based controls or an on screen interface). 2. The command is passed to the system through standard internet protocol. 3. The system identifies the robot paired to the specific player. 4. The movement command is transmitted to the robot through the Bluetooth connection. 5. The robot receives the movement command and performs the specified movement. |
| **Post Condition** | The movement command issued by the player is carried out by the paired robot. |

|  |  |
| --- | --- |
| **Use Case Name** | **Fire virtual projectile** |
| **Description** | A mobile player issues a command to fire a virtual projectile, which is created and simulated in the system. |
| **Precondition** | A game is in progress.  The player is currently paired to a robot.  The active game mode is “Tank Simulation”. |
| **Primary Actor** | Player |
| **Secondary Actors** | None |
| **Dependencies** | None |
| **Basic Flow** | 1. The player issues a command to fire a virtual projectile. 2. The command is passed to the system through standard internet protocol. 3. The system identifies the robot paired to the specific player. 4. The system determines the current position and heading of the paired robot. 5. The system creates and simulates a new virtual projectile with the same position and heading. |
| **Post Condition** | A virtual projectile is simulated originating from the player’s paired robot’s position. |

|  |  |
| --- | --- |
| **Use Case Name** | **End game** |
| **Description** | A game in progress is ended and all connected mobile users are returned to the game lobby.  Any paired robots are freed and considered unpaired. |
| **Precondition** | A game is in progress. |
| **Primary Actor** | Player |
| **Secondary Actors** | Robot, Mobile User |
| **Dependencies** | None |
| **Basic Flow** | 1. The end of a game is triggered when directly requested by a player, when connection with an active robot is lost, or when the ending conditions of the active game type are met. 2. The system displays a win/loss message to active players if appropriate. 3. Remote control of robots is disabled (all pairing between robots and players is removed). 4. The system directs the robots to their original starting positions. 5. The augmented video feed to mobile clients is disabled. 6. The system changes state to reflect that no game is currently in progress. 7. The system clears all virtual game state is to prepare for the next game. 8. The server lobby is displayed to all connected mobile clients. |
| **Post Condition** | A game is no longer in progress.  The server lobby is visible to all connected mobile users.  All robots are free (not paired to any player). |