Input Feedback Refactoring Specification

Initial Proposal: Jeffery Myers

Rev1 5/1/2017, initial proposal

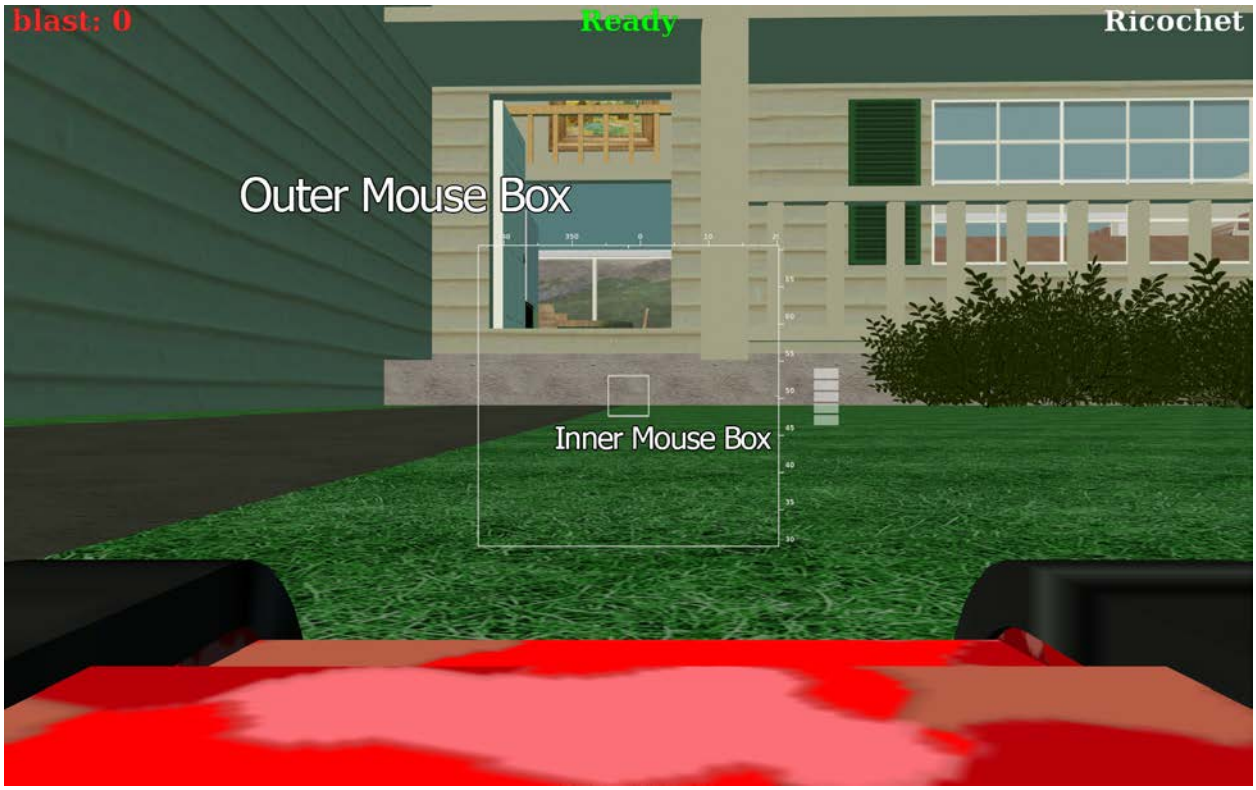
1 PURPOSE

This project is designed to replace the current input feedback system, the visual display to the player about the state of his or her input devices.

The current software supports 3 different input systems.

- Keyboard
- Mouse
- Joystick

Only the mouse system provides visual feedback. This feedback is provided by the use of the standard operation system cursor and its positions relative to a pair of “mouse boxes” on the screen. The inner mouse box represents an input factor of zero for an axis, and the outer mouse box is 100% speed in the axis.



1.1 PROBLEMS WITH THE CURRENT SOLUTION

- Confusion about what the mouse cursor does.
    - Users have been trained for decades that mouse cursors are for clicking, yet in BZFlag nothing is clickable.
    - Because the system allows driving while menus are up, a cursor is shown over the menus. This makes the menus seem clickable, but they are not. This is a barrier to entry for new players, and makes the product look incomplete/buggy.
- Confusing limits
    - The inner/outer box paradigm is not clear. Many users think the top of the screen is 100%
- Resolution Dependent
    - By using a screen space cursor and box system, mouse input is limited by the number of pixels on the screen. Systems with more pixels have more input granularity.
- Lack of a “Sensitivity” Setting
    - Because the system is screen based, there is no simple way to adjust mouse sensitively.
- No aiming sight.
    - The center area of the mouse box can function as a crosshair, but it’s size also affects the dead zone for mouse movement.
- No display of other input methods.
    - Joysticks and keyboards do not get a similar display.

2 PROPOSED SOLUTION

2.1 UNIFIED INPUT FEEDBACK SYSTEM

A replacement for this system would include a single unified GUI item that provided feedback for all input types, and be device/resolution independent.

2.1.1 Remove the mouse cursor

Since there is nothing in the BZFlag GUI that is clickable, there is no reason to show the OS mouse cursor. The cursor will be hidden as soon as the game starts up and initializes its OpenGL Context. The cursor will be restored whenever the game window loses focus.

2.1.2 Remove Mouse-box system.

The mouse-box system is no needed, and will be removed. New GUI elements will take its place. All mouse box related BZDB data will be removed/ignored. This includes removing the current compass, altimeter, and shot bars. These features will be incorporated into the new GUI system.

2.1.3 Add Mouse sensitivity and dead zone user settings.

Add BZDB items for an overall mouse sensitivity and dead zone areas, just like exist for joysticks. Provide separate settings for each mouse axis (X and Y).

2.1.4 Add Aiming Crosshair

An aiming crosshair will be added to the GUI. This crosshair will be configurable in size (and maybe style?). It will always be centered on the screen where the shot exits the view plane. Its existence is completely independent form all input systems.

2.1.5 Add Input Feeback Hud item

This GUI element will contain the Compass, altimeter, and shot reload display, as well as feedback for the positions of each input axis. This GUI item will be user sizeable based on BZDB vars.

Compass, altimeter, and shot display will function identical to the current code, they are just repositioned/scaled based on the new GUI settings.

Each input axis shows 4 items of data.

- The positive and negative limits of the axis.
- The dead-zone of the axis.
- The current input position along the axis
    - This is the desired speed as provided by the user.
- The actual speed of the tank along the axis.
    - This takes tank acceleration into account, and will lag behind the actual input as the tank catches up.

2.1.5.1 Diagram

Below is a rough diagram of the proposed GUI. Elements in this image are larger than they should probably be in the final implementation, just to show the various parts of the GUI.

2.1.6 Functionality

Each axis shown has a maximum value, a current desired value, and the current actual value.
This display is functional for all input types, and provides consistent feedback to users, about how their tank is preforming relative to the desired control positions.

The speed that the desired input changes is controlled by the sensitivity settings for the current input method. More sensitive controls will change desired positions more quickly. Input systems without sensitivity options, such as keyboard, will instantly change desired input positions. The actual speed will change only based on the tank acceleration settings. This provides accurate feedback to the player.

2.2 Implementation Details

The implementation of this should be pretty straight forward. The same GUT code that is drawing the mouse boxes can be modified to draw the new GUI element. Some changes to the mouse processing will need to be done, but it should be similar to the code used for joysticks. Since all platforms now use some version of SDL, there may be SDL tools to assist with this.

2.3 Alternative Implementation

If a simpler initial implementation is desired, to minimize “GUI SHOCK” to the user base, the system can be implemented in multiple phases, with a simpler initial system that looks more like the legacy GUI.

This setup would keep the existing screen based mouse boxes so all existing settings would translate over directly (keeping the game feel). The only difference would be showing small arrows on the box to indicate where the mouse axes are.

3 Open Issues

3.1 What drawing options are needed for new GUI

What kind of controls will people want for the new GUI? Thickness? Colors? Transparency? What should the defaults be?