Production Brief – Bullshit Pong!

## Premise

***Bullshit Pong*** is a pong game developed for educational purposes with the intent to confuse and piss off audiences. Its technological goals are to showcase a fundamental understanding of game development and programming concepts.

## Game Design

Bullshit Pong is a variant of the classic *Pong* <http://www.ponggame.org/>. In this variant, the players are represented by paddles, which can move up and down vertically, positioned at the left and right sides of the screen. The ball however is wiggling and moves very fast. The players simply bounce the ball back and forth until someone reaches 7 points.

### The basic rules of the game are as follows;

* The game space consists of 4 boundaries around a rectangular playing field.
* Players begin the game on the left and right sides of the screen.
* The game begins with the ball wiggling from the center of the field to the player's paddle.
* Whenever the ball collides with the side walls, the player's score should increase and the ball should respawn in the middle. The score should also be visible on the screen.
* Whenever the ball collides with the top and bottom walls, it should bounce back in a pseudo-random direction.
* There are 2 game levels the second level with inverted controls to mess with the players.
* The game ends when one of the player's scores reaches 7.

## Technical Decisions

The rendering, input management, and timing will be handled by SFW, a simple library developed for educational purposes by Ginger Farrar ginger.farrar@students.aie.edu.au. GitHub and Visual Studio are the primary software stack.

## Production Goals

Week 1; **Game State:** Complete implementation of the previously mentioned rules within a real-time graphical environment.

Week 2; **Application State Management:** Encapsulation of game objects and state to accommodate an application state management scheme. The states will include, *at the very least* a; Splash Screen, Main Menu, Game State, Victory State, and an Exit state.

Week 3; **Feature Development:** Extension of the above rules in *at least two* unique ways. Implementation of *at least* one additional application state.