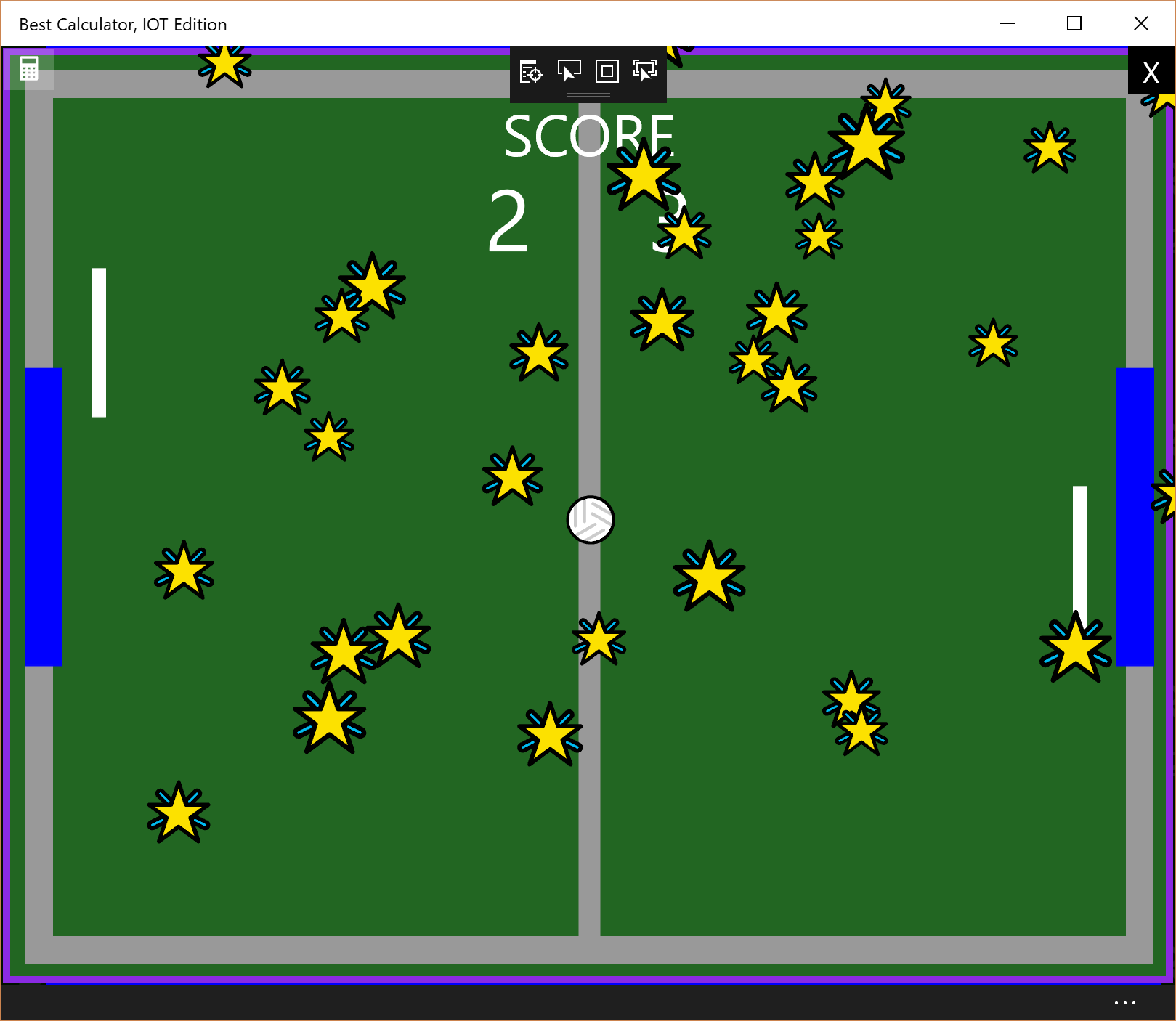
# Making a PONG-like game in BC BASIC

In less than 7 hours, I made a PONG-like two-person game using the BC BASIC programming that is built into the Best Calculator (available from the Windows store).



*Playing a game of PONG*

BC BASIC is similar to the classic retro-BASIC for Sinclair, BBC BASIC, Atari, Commodore and Apple micro-computers from the 80’s but with some modern additions. The language supports more powerful programming constructs without the constrains of the old micro-computer BASICS: it has proper functions, programming without line numbers, and multi-line IF statements. It also supports modern hardware with simple high-resolution graphics, Unicode, and mouse input. The paid version also includes simple Bluetooth, HTTP and sensor support. At the same time, the system is designed to have very low barriers to entry: writing your first BASIC program, once you are running the Best Calculator app, takes exactly three mouse click (to enter the overall BC BASIC system, to create a program, and to run it). The main BC BASIC language is kept small so it’s faster to read the manual. And, of course, a full 300-plus page manual comes as a PDF with the program.



*Scoring a goal triggers the goal animation*

Making the PONG program went smoothly. I use a trivial “list of work items to be tacked” that I keep in a Word table. The first column tracks the time for each work item and the second column gives a few details on the work items. Work items for large projects are often kept pure, but for a short one-person project it’s quicker for each work item to have a main feature to add or bug to fix (like, “adding spin to the ball”) plus a series of small improvements and bugs that become evident while working on the main feature.

# What went well/what went badly

The BC BASIC system language was easy to work with. Using Unicode for many of the graphics worked well; the goal and winning animations were trivial to make and look more polished than the 30 minutes they took to make. There’s an easy-to-use intersection system in BC BASIC that made the hit detection simple although I wish that it provided richer information about the hit (notably, which side was hit, because that would make the “physics” uniform and would allow for some more interesting game additions).

Some of the drawbacks are that the GLOBAL command in the functions was clumsy and error prone. The editor, which is designed for very small (50 lines or less) programs was clumsy when dealing with the much longer PONG program. The lack of a debugger was very frustrating at times.

# How to play

The left player controls are the **A for up** and **Z for down** controls. The right player controls are similar, with **K for up** and **M for down**. There isn’t a back-and-forth or tilt control. If you hit the ball after moving a paddle, you will add some “spin” to the ball and it won’t just bounce back. To score a point, the ball has to go into the other player’s goal (the blue rectangles on the edge of the screen). Games are played to 11 points, but you have to win by two points. After each game, press **N for a new** game or **X to exit**.

There are a couple of debugging controls Press G to trigger a goal animation, or W or E for the winner animation. S will pause the ball and D and C will add “spin” to the ball.

*The winning animation. The trophy is placed on the winner’s side*

# Game Program Structure

The game is structured with a *preamble* to set up the global variables followed by setting up the *SetInterval* timer that will call the *Frame()* function about every 50 milliseconds. The game then runs FOREVER (which usually means pressing the X key after a player has won)

The key global variables are:

The **g** (graphics) variable is the commonly-used name used for the global variable that holds the grpahics screen in many BC BASIC programs. All of the other objects are created from the g object.

The **ball** is a graphic text object; it’s position is directly controlled via the ball.CX and ball.CY (center-X and center-Y) positions. The ball speed is controlled with two global variabless **dx** and **dy** (delta-x and delta-y). Each **Frame** call, the ball position is updated by dx and dy. Additionally, when the ball hits a paddle the dy value is updated with a **p1dy** or **p2dy** (player 1 or 2 delta y) “spin” value which is set by the movement of the paddles.

The ball will bounce off the sides. The there four **side** variables (side1 through side4). When ball.Intersect (side<n>) is 1, that means that the ball hit the side. In all cases the point of intersection is calculated and the ball’s dx or dy values are flipped and the cx and cy values are updated.

The ball can also intersect the two paddles (**paddle1** and **paddle2**) (with similar results) or the goals **goa1** and **goal2** (which triggers a goal animation). The player scores are the **score1** and **score2** variables and are displayed on the screen with the **scoretext1** and **scoretext2** variables. These are both graphic texts that are set up when the program starts. Their text value is updated with the Text property (e.g., scoretext1.Text = 10 will set the left player score text to read “10”).

To keep the game states straight, there’s a **state** variable with value r=running s=stopped g=goal animation or w=winning animation. Yes, it’s just a plain string because BC BASIC doesn’t include an enumeration type. For the goal and winning animation there’s a countdown timer; when the countdown ends, the animation is complete.

The animations consist of a **flowers** or **stars** array of flower or star graphical text objects. They are set up ahead of time and given an opacity of 0 (not visible). When an AnimationStart() function is called, the flowers or stars are set to Opacity=1 and given a new, random position on the screen. Each Frame, the animation functions are called to update the screen position (each star or flower is moved by a random amount up and over). The **trophy** graphic text holds the Unicode Trophy character; it’s set to be centers (trophy.Alignment = “CC”); changing its FontSize makes it animate to be larger.

One last point of interest: BC BASIC includes a unit test for the **IsWinner**(score1, score2) function. The test function is an ordinary function called TEST\_IsWinner that takes no parameters. Inside there are a series of calls to **ASSERT(**expression) where the expression must be an *equality* expression. To run the tests, simply click the checkmark on the editor; that will run all of the test functions and tell you which ones worked and which did not.

By making the test framework trivial to run –absolutely no configuration required, and running the tests requires just a single mouse click – inexperienced programmers are guided into a best practice.

# Timeline for making the game

Following is the simple system for tracking the coding for the game. The table started with just the first entry which got the game to a point where there was “something” on the screen. After that, new lines were added to the table as needed. Each item in the table was done in order (but that’s not a hard-and-fast rule).

|  |  |
| --- | --- |
| 2:47 to 4:08  81m | Started game code. Create the sides and ball and made the ball bounce.  Fixed bug in the BASIC concept of the screen size (it didn’t take into account the padding and border size). This will be fixed in BC BASIC version 3.16  Had to kill and restart Visual Studio multiple times. |
| 4:10 to 4:22  11m | Added paddles and INKEY$ for them but not the ball-hit-paddle code |
| 4:23 to 5:04  41m | Added paddle-bit detection  Had more issues with Visual Studio hanging and crashing. |
| 1:26 to 1:53  27m | Added goals and score areas |
| 2:15 to 2:45  30m | Added ‘s’ command to stop the action and dump variables  Added animation for goal |
| 2:50 to 3:08  18m | Bug: if you don’t move the paddles, they aren’t seen by the ball-bounce function  Also made the stars look better, and shifted the paddles a little to make it look nicer |
| 8:08 to 9:52  104m | Add “spin” for ball (otherwise, after a goal, the game is just back and forth!)  This took too long to fix bugs.  Improved paddle handling so that paddles don’t go off screen |
| ??? | When the game starts from a button, need to get text focus (whoops!)  And click on full screen window after losing focus should make INKEY$ work  [for these two: INKEY$ now forces focus]  Should not need CLS PRINT to make the full screen window pop up |
| 10m | Add “hit” sounds and goal sounds  Use Melodic Tom=119, octave 4, notes F and G |
| ???  45m | Make a winner   * Scoring is to 11 points and must win by two IsWinner(player1, player2)🡪 0 1 or 2 where 0 is no winner yet * Show a nice trophy (🏆 🍾) on one side or the other * And lots of pop-up flowers 🌷 🌸 🌹 🌺 🍁 🌻 🌼🍒 🏵 💥💮 * Or fireworks 🎆 🎇 🎈 🎉 🎊 |
| 5:09-5:55  46m | After a game, set up for the new game and fix issues. Lots of fit-and-finish punchlist type items.   * Clear out the stars * Set the winning score back to 11 * Fix the star size (they are too small and are being clipped) * Add center line * Comment out the hit line * Add the word SCORE * UR on right side is all goal? (the ball was not centered and the bounding box was too large) * Type space bar to play again (no, it has to be N) |