Article
Timothy Leavitt · Dec 16, 2019 6m read

Terminal tricks for fast display and fun formatting (Advent of Code 2019, Day 13)

I've been having a blast with the Advent of Code puzzles this year - though I'll be heading into a busy span of time with family soon and will probably drop off toward the end. (At least, that's what always seems to happen - it's a good thing, though!)

I had a whole plane ride to play around with Day 13 and wanted to share some fun terminal tricks. I'm not planning to post my solutions on GitHub until the end of the year, but this is just visualization fun - not relevant to solving your own puzzle input (although visualization definitely helps for debugging) 

In my initial visualization, I was disappointed by how slow it was and how choppy it looked. This was because my initial implementation cleared the screen and wrote out the entire state of the game each frame. This is silly, because only a few positions change each time the ball moves. Less critically, it also was a bit boring and dissatisfying having letters to represent the different parts of the game. That's where ANSI escape codes come to the rescue!

For this exercise, there are two relevant capabilities in the ASCI escape code toolbox:

- Adjusting the X and Y position of the cursor
- Changing the background/foreground color

There are plenty of good references and writeups on the internet for doing this in other technologies, so I'm just going to link to them and show my code + results rather than explaining everything in detail. Other than the Wikipedia article linked above, here are some good sources:

- http://ascii-table.com/ansi-escape-sequences.php is another handy reference for all the different codes/capabilities
- https://docs.intersystems.com/iris20194/csp/docbook/DocBook.UI.Page.cls?KEY=RCOSvx and https://docs.intersystems.com/iris20194/csp/docbook/Doc.View.cls?KEY=RCOSvy cover the $X and $Y special variables; particularly, note:
    - "ANSI standard control sequences (such as escape sequences) that the device acts on but does not output can produce a discrepancy between the $X and $Y values and the true cursor position. To avoid this problem, use the WRITE * statement and specify the ASCII value of each character in the string."
    - "As a rule, after any escape sequence that explicitly moves the cursor, you should update $X and $Y to reflect the actual cursor position."

It's also worth noting that, while $X and $Y are 0-based, the ANSI escape sequences to update $X and $Y are 1-based.

As for how this looks in ObjectScript, here's my code:

```
ClassMethod AdvancedDisplay(ByRef map, score, Output printState)
{
    If '$Data(printState) {
```

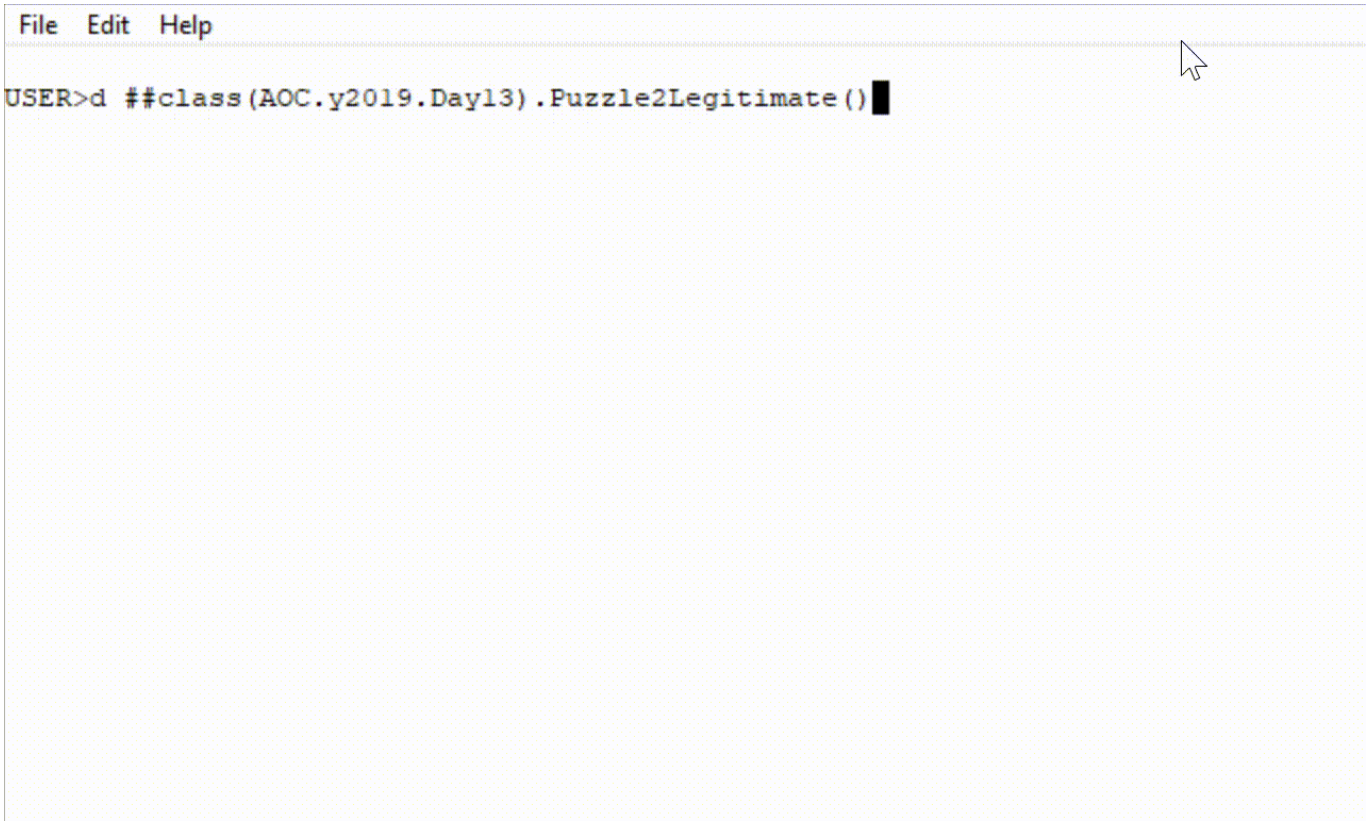
```
        // Clear out the screen the first time this method is called
        Write #
    }
    If (score '= $Get(printState("score"))) {
        // Move the cursor to the top left
        Write *27,*91,1,*59,1,*72
        Set $X = 0
        Set $Y = 0
        Write "SCORE: ",score,!
        If $Data(printState) {
            Set printState("score") = score
        }
    }

    Set y = ""
    For {
        Set y = $Order(map(y))
        If (y = "") {
            Quit
        }
        Set x = ""
        For {
            Set x = $Order(map(y,x),1,data)
            If (x = "") {
                Quit
            }

            // If the map doesn't match the previous version at this position, re-
render this position.
            If (data '= $Get(printState(y,x))) && ($Data(printState) || (data '= 0))
{
                If (data = 1) || (data = 2) || (data = 4) {
                    // Set background to black (border), red (brick), or green (ball)
                    Write *27,*91,(10 /*background*/ + $Case(data,
                            1:30/*black*/,
                            2:31/*red*/,
                            4:32/*green*/)),*109
                } Else {
                    // Reset all colors
                    Write *27,*91,*0,*109
                }

                Write *27,*91,y+2 /*extra line for score*/,*59,x+1,*72
                Set $Y = y+1
                Set $X = x

                Write $Case(data,3:"_",:" ")

                If $Data(printState) {
                    Set printState(y,x) = data
                }
            }
        }
    }

    // Reset all colors
    Write *27,*91,*0,*109

    // Move the cursor to the bottom of the game
```

```
    Set realY = $Order(map(""),-1)
    Write *27,*91,realY+3,*59,1,*72
    Set $Y = realY + 2
    Set $X = 0

    // Reset all colors
    Write *27,*91,*0,*109

    // Store print state (updated incrementally on future calls to this method)
    If '$Data(printState) {
        Merge printState = map
    }
    Set printState("score") = score
}
```

Outside of this, there are a few other commands that hide/show the cursor:

```
Write *27,*91,"?",25,"l" // Hides the cursor
Write *27,*91,"?",25,"h" // Shows the cursor
```

And, saving the best for last, here's the result:



It's so fast that you almost need to put a HANG in there to slow it down!

#Code Snippet #ObjectScript #Terminal #Caché #InterSystems IRIS

Source URL:https://community.intersystems.com/post/terminal-tricks-fast-display-and-fun-formatting-advent-code-2019-day-13