Assembly Project: Breakout

Aleksey Panas

April 10, 2023

1 Instruction and Summary

1. Which milestones were implemented?

   Features include:

   (a) Precise arbitrary velocity ball movement and collision computation. Computes a unit vector under an 8 bit shift for float accuracy and moves ball at most 1-pixel at a time to detect accurate collisions. Once velocity value reached, updates ball display

   (b) Paddle bounce angle changes

   (c) Bricks can have arbitrary top left corner, width/height color, and number of lives. Brick color fades with each hit based on fraction of its remaining lives

   (d) Walls can also have an arbitrary size, color, and location

   (e) 5 regular levels with looping in play mode

   (f) player lives ingame

   (g) score and highscore

   (h) menu with full level selection and locked levels until beaten

   (i) Up to 5 custom levels

   (j) Full editor to build custom levels

   (k) Pausing

2. How to view the game:

   (a) Open and run the file in Saturn (doesn't work in MARS)

   (b) Use ASWD to navigate menus

   (c) Use Spacebar to press buttons, launch the ball, and continue from game over or win screen

   (d) Use R to return to previous menu, or to menu from editor

   (e) Use Square Brackets to change selected RGB or lives value in editor

   (f) Use Q and E to increment or decrement the selected value at the top in editor

   (g) Use ASWD to move cursor in editor

   (h) Use 1 and 2 to set corner 1 and corner 2 of the selection in the editor

   (i) Use X to delete the item at the cursor in the editor

   (j) Use B to place a brick and V to place a wall in the editor. Cannot overlap other game objects

   (k) High score and level unlocking only happens in PLAY MODE, by hitting play (ie selecting a later level from level selection wont count towards score or unlocks)

The project took a total of 74 hours to complete over the course of 2 weeks. The final file has 6200 lines of assembly code. I improved as I went so many areas could be shortened and outsourced into functions. Still, I planned functions from the beginning and ended up using the stack extensively.

Some functions I implemented include arbitrary collision between two rectangles, bitmap renderer, arbitrary integer renderer, filled and hollow rectangle renderers, vector to screen address converter, two points to top left width height format converter, and countless other helpers.

To navigate the game, I used an underlying state machine. The main loop is very thin and simply delegates to the corresponding state handler. A "dirty" flag is used to trigger a state change and must be set to 1 along with state change parameters.

However, not everything is limited to assembly. To help construct the bitmaps, I built a quick 200-line pygame script in python to be able to draw bitmaps and convert them into the format necessary to directly paste into the .data section. A second script was used to help preserve levels in an easier format and then convert to the .data format

To test certain utility functions in isolation, I created additional asm files for unit testing.

Submitted along with this report and the assembly file is a PDF document of my digital notebook where I wrote down most of my code planning, as well as outlines of every struct used in the code. Without this, there is no way the code will be readable

The entire project is on my github at the following link: https://github.com/AlekseyPanas/assembly-breakout