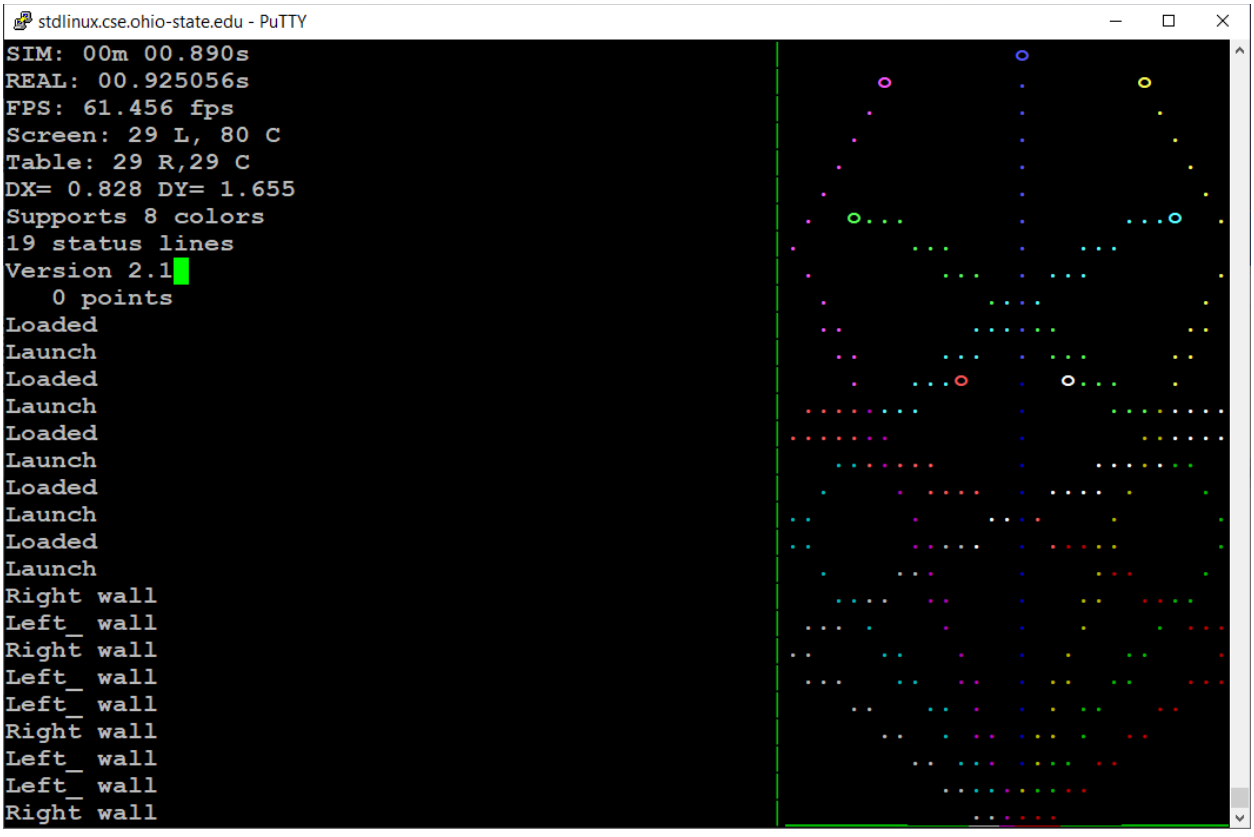
Overall Lab 3 SP23

Contents



As the above picture shows, in lab 3 we will run all of the balls simultaneously. We will use 2 linked lists to keep track of them and a single structure to keep track of the lists and everything else. Most of the

basic lab 2 simulation is correct or nearly correct. Our code reads balls, transforms the ball data, and runs the table.

See the design-by-comment document for exact details of the sequence of the core simulation loop. Note the following items need to be in the correct order in the main simulation loop:

- Output the existing state of the world
- Change the world:
    - Update the simulation time by DELTA_T
    - Move things
    - Constrain things (walls then flippers)
    - Anything that went off the table is removed from the in-play list and added to the off-table list

Differences with Lab 2

Lab 3 does not wait for one ball to go off the table before reading the next one and putting it on the table

Lab 3 stores our balls in dynamically allocated structures so we can have as many in the program as we need at any particular time.

We need a second structure to hold all of the simulation data, including a linked list for balls in play and another list for balls that are off the table. The simulation structure also holds the simulated elapsed time and the score.

The simulation runs until the in-play list is empty.

All dynamic memory must be freed before the program exits.

The linked list functionality is supplied to you in lab 3 – see piazza.

Things that are the same

- Divide your code into files as before – you might add a file for callback functions that have no other home. Dynamic memory gets its own file.
- Motion computations are the same

Using Instructor Supplied Lab Code

You can base your lab on your own code or you can use the instructor supplied code.

Copyrights

You must retain the copyright notices in the C code files. Any header files generated by the headers.sh script will have your name in those headers and that is fine. If you delete the copyright notice and turn in the file with your name on it, it is a CoAM offence (misrepresenting another person's work as your

own). If you delete the copyright notice and don't put your name in the C code file, you get a **zero** on the lab because all files turned in need the author's name in them.

Changes

If you make changes to a file, add your name at the top of the file under the existing copyright notice. You don't have to add a copyright notice of your own, having your name in there is enough for this class. If you use a file without making any changes to it, it can be turned in as it is.

Any function that you change needs a comment above that function where you describe what you changed.

Any function that lacks a "one purpose" comment needs to have one added or your lab will lose points for not having the required comment.

Make sure that the makefile has your name in it and comment the lines for your prototypes.

Additions

Added functions need to be marked as yours. Added files need all of the normal markings.

Copy – Paste

Code you extract from the supplied code needs to be marked so that there is no doubt as to the author. This is useful in cases where one or two function in a file are useful but you don't want the entire file.

Please get it right

The supplied code is there to help you. Note that the infractions for misusing the code are steep; a zero on the lab or a possible CoAM case. If you have any questions at all, please ask. Note that you are not allowed to put this code on a public server at this time. You would need explicit permission to do that.