Project 1: FireFighter Game

Goals

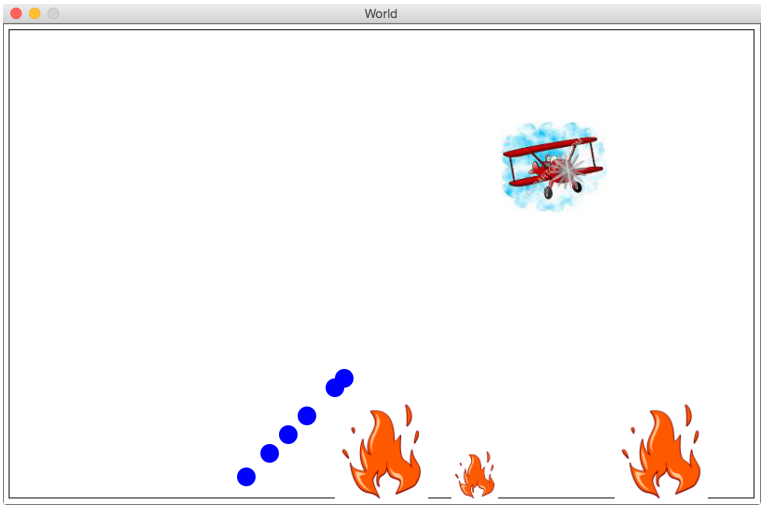
- To make sure you can write programs using combinations of lists and structures.
- To practice using the design recipe to organize a nontrivial program.

It goes without saying (we hope), but you must use the Design Recipe for all functions. In particular,

- Don't forget to write a signature, purpose statement, and unit tests for every function.
- Write the tests before you write the function definition.
- Use templates to organize the function body
- Break up functions into smaller functions if they become too complex. ONE FUNCTION ONE PURPOSE. You will need many auxiliary/helper functions.
- Give a Data Definition for every data type you use.

The Assignment

Your job is to write a simple interactive game in which the player drops water from a plane onto fires on the ground; the goal of the game is to put out all of the fires by dropping enough water on them. A sample appears below:



The plane flies continuously back and forth across the screen (ie, the user does not control whether the plane moves). The player can change the direction of the plane by pressing the left and right arrow keys. The player can also drop water from the plane by pressing the down arrow key. Each fire has an intensity: when a drop of water hits a fire,

the intensity decreases. If a fire burns without water hitting it, its intensity increases. A fire is extinguished when its intensity reaches zero.

While we have dramatically simplified many aspects of the the game play, it still will be by far the largest program you have constructed in class. Strict adherence to the Design Recipe is required to keep everything straight. In Project 1 we still provide a large amount of structured help, but be aware than, unlike a lab, most design decisions will be up to you and your partner. Beyond the first part of the project, we will not be telling you "develop this function, then that one, then...".

We clarify the minimum functionality in the game required for an "A", however we also point out many opportunities for extra credit, as this is a very extensible premise.

Game Components and Gameplay

As you can see, there are three main components of the game: the plane, the water drops, and the fires.

The Plane

The plane direction is controlled by the player using the left and right arrow keys. The plane is constantly moving, and will switch direction when it reaches either end of the screen. The view of the plane should visually indicate the direction in which it is flying.

The Water Drops

Water drops fall at a constant speed directly downward. Water drops are released from the plane, at the current position, when the down arrow key is pressed.

The Fires

The game should start with 3 fires of medium intensity. Over time, fires should grow larger and new fires may start. As they are hit by water, fires will reduce intensity. The view of fires should indicate visually at least 3 sizes of fire (small, medium, large). The game is lost if a fire reaches maximum intensity; the game is won if all fires are put out.

Software Engineering Strategy

If you and your partner prefer to work separately, work out and agree on the data definitions first, then proceed to work on the rest of the code (together or independently). In a professional setting, one person or team is responsible for each data definition ***and all the functions that operate on that data definition***. Function signatures become literal *contracts* between you and your teammates.

Implement the game in stages ("increments"), not all at once (see ***Grading***).

Hints

- You will be graded primarily on the structure of your code, not on efficiency. *Don't worry about efficiency.* Your computations to check whether a water drop has hit a fire need to be reasonable, but do not need to be exact (i.e., don't waste time on getting these expressions correct down to the pixel). If the resulting game play looks plausible, we won't care about the precise formula you used.
- You may simply specify a `stop-when?` function to stop the game when all fires are extinguished.
- You may use whatever images you wish for the pieces of the game (the ones we used are attached). For example, one student many years ago made this:

- Remove extinguished fires from the world: certain aspects of the game will be much easier to program if you do this.
- Raise the intensity of the fire at a much lower rate than you drop it (or the game becomes unplayable). Our prototype raised intensity at 1/100 the rate of lowering it, for example. Use real numbers rather than integers for this.

Grading

When we grade your projects, we are primarily looking at the functionality you have implemented.

- To earn a D, your project must implement the plane including its movement and facing (This is Lab 5.)
- To earn a C, your project must implement everything needed to earn a D plus adding water drops to the game

- To earn a B, your project must implement everything needed to earn a C plus 3 initial fires that decrease in intensity when water hits (and the game stops when these three fires are out)
- To earn a A, your project must implement everything needed to earn a C plus both putting out fires, and having fires intensify over time, and the possibility of new fires starting (i.e., the complete game).

Once we determine your starting grade, we'll look at the code and deduct points for the usual problems: missing signatures, missing tests, poor formatting, etc.

You will get more credit for a fully-working, well-designed solution to part of the game than for a non-working solution to the whole game.

Extra Credit

This project lends itself to many extensions. If you care about exactly how much credit we will award, please check with us first. In general we won't award more than 30 points extra.

Here are some specific ideas:

- Add Wind to blow the water (both x and y position changes); indicate current wind speed and direction on the View.
- New types of "fires" with different behaviors (e.g., fire dragon moves back and forth to avoid drops and starts new fires, Godzilla fires back at the plane, etc.)
- Additional items to interact with (power-ups e.g., limited charges of chemical flame retardants that have larger effect than water, fuel boosters that make the plane move faster, etc.) that are picked up as the plane flies over them (only on screen for a short time, dropped behind plane to force change of direction, etc.)
- Multiple levels of difficulty
- Scoring, and High score tracking (see `2htdp/batch-io` for writing/reading a score to a file)

What to turn in

Turn in a single file project1.rkt containing all code and documentation for this assignment. Make sure that both students' names are in a comment at the top of the file. Projects may be submitted late, at a penalty of 15% of your grade per day. (Yes, weekends count).