Comp160
Project 1
Spring 2018

The Project

For your first project you'll be completing and adding to the virtual pet program as described in Section 5.11. The extensions are broken down into two tiers where each tier adds more complexity and as a result opens up a higher possible grade for the project. Extensions should be carried out in the order they are presented below and are meant to be incremental feature additions to the program like you might encounter in software development. The end result is a single program. Once you complete one extension, you simply modify and revise that program to incorporate the next extension.

Tier 0 Project

To complete the basic, no frills virtual pet program you must finish exercises 88–91 in the text. These exercises combine the walking cat and the happiness meter from our prior work and then extend this program by making it stop when pet happiness reaches 0 and making the cat walks back and forth rather than wrapping around the screen.

Tier 1 Extensions

The first four extensions to the pet project can be done without changing the underlying structure of *VCat*. Some logical changes will be made and reflected in revised data definitions, but the structure definition itself does not change.

1. *Low Happiness Warning*

   Change your program so that a thin red border is drawn around the whole window whenever the cat's happiness is 10 or less. Notice this changes the happiness of a pet to an itemization of intervals. Be certain to update your VCat data definitions accordingly.

2. *More Happiness Level Feedback*

   The previous extension lets the user know that the game is about to end. Give them a bit more warning by also changing the color of the happiness bar as key happiness levels are reached. If happiness ranges from 50 to 100, then the bar should be green. If happiness is between 25 and 50, then make the happiness bar yellow.

Finally, if the happiness is less than 25 make the bar red. Notice that this further refines your happiness intervals and requires another revision to the data definition.

3. *Realistic Petting*

   Now let's make our cat more realistic. Cats will often turn around as you pet them. Update your program so that the cat changes directions whenever it is pet.

4. *Moody Movement*

   To add further realism to our virtual cat, let's make the cat's movement dependent on its happiness. Change the program so that the cat also changes directions whenever the color of the happiness bar changes. This provides more feedback to the player and simulates the implied mood shifts of the cat as movement changes as well.

Tier 2 Extensions

The fifth and sixth extensions require more substantive additions to the program than the previous four. You'll need to change the structure definition for VCat and work with parts of world programming that have not yet been covered in class.

5. *More Petting!*

   It is very common for programmers to be in a position where they pick up something new but familiar by sorting through the documentation and reference material for libraries. Section 3.6 describes how world programs handle mouse events by providing basic signatures, purpose statements, templates, and examples. This programmer's documentation interface is all you need to start working with mouse events in world programs.

   Change your virtual pet program so that clicking the mouse on the cat is a second way to pet the cat. The end result should be the same as petting via the keyboard: the cat gets happier and changes direction.

6. *Cats are Nocturnal*

   Cats tend to be more active at night then they are during the day. Let's try simulating this by adding a time component to our virtual world. Time should advance by one hour every tick. To simplify things we'll work in terms of 24 hour time. Let's call daytime 8 to 18, night 20 to 23 and 0 to 6. Dawn and Dusk occur from 6 to 8 and 18 to 20 respectively. To simulate the cat's preference for night time activity, we'll make petting and feeding the cat increase

happiness by twice their normal amount if it's night time. To indicate to the player what time it is we'll change the background colors. During the day the background should be white, at dawn and dusk it should be light gray, and during the night it should be black or some darker shade of gray.

Adding time to our virtual world will require us to add a new variable to the world state. You'll need to revise not only the VCat data definition but the BSL Racket structure definition for VCat as well. Unlike our previous extensions, this change is not *backwards compatible*, meaning it is not compatible with previous versions of your program[1]. It's going to break your program and you'll need to go and revise a good number of functions and tests to account for the change. Programmers attempt to use good design practices and planning to avoid these kinds of changes, but they are often unavoidable because predicting the future is hard. Who knows what new feature you'll need or want down the road?

[1] This is also true of the addition of directionality to the cat's movement.

Grading

Your grade will be determined by two factors: your progress through the tiers determines the letter grade range that is open to you and the quality of your work will determine where you land within that range.

Tiers and Grade Ranges

In order to get a C- or better you must complete the basic, tier 0 program. From there you must complete the extensions in the order they are listed above. Progress beyond the C level is determined by the tier into which your program falls as listed below. Assuming that your code meets the expected quality standards, progress within the tier will increase your grade within the tier's associated grade range; You can expect to achieve a higher B by completing the first three extensions than if you only complete the first extension.

Tier	Grade Range
0	C- to C+
1	B- to B+
2	A- to A

Progress is measured by results not intension. Results are measured by functioning code. You need to complete the extension, and all prior extensions, such that the program will run with no real significant problems. It should not crash and it should not behave in unexpected ways. In the event that you cannot complete an extension then you should submit something that runs and does some part

of the extension rather than submit something that doesn't run but has code that attempted to deliver on the extension's requirements. Code that doesn't run or that causes the program to crash in predictable ways is not the type of code you want to submit. It is not what counts for partially correct in programming. Work the How to Design Programs process and build up your program incrementally such that you can always revert a non-functioning program back to a working one without too much work.

Quality

A high quality program exhibits all the earmarks of intensional, systematic design and good programming style. Program data is appropriately defined. Functions are well documented. Incomplete functions are stubbed as opposed to commented out. Complete functions typically exhibit template structure when applicable. Complete and incomplete functions have a full set of tests. Signatures, purpose statements, tests and function definitions are all appropriately placed. Function and variable names are helpful and meaningful. Purpose statements are specific and concrete. Indentation of code follows expected Racket standards[2]. Lines are terminated to avoid print wrapping. Comments are used to aid the human reader. Whitespace is used to break up logical blocks of definitions. The degree to which you meet these standards, along with progress within the tier, determines where you fall within the grade range associated with your tier. It is entirely possible the poorly designed and styled code the completes three extensions can get a lower grade than a well designed and styled program that completes one or two extensions. Do not short yourself points by writing sloppy code.

[2] It's styled like the code in the book.

Important Dates

Date	What's Happening
Monday 3/19	Open Lab Time to For Project Work
Tuesday 3/20	Open Lab Time to For Project Work
Monday 3/26	Printed Copy of Code Covering at least exercise 89 due in class.
Wednesday 4/4	Code due in class.