Project 1: Virtual zoo

Due: 1 Nov 2010

In this project, you'll build a 'virtual zoo', a small collection of virtual pets that require care and upkeep. This is somewhat larger than anything you've written so far, so the overall timeframe for the project is two and a half weeks, but there are a few intermediate checkpoints to keep everything on track. Just as we do on a smaller scale, writing a complex function by breaking it into a few smaller, simpler ones, we will attack the larger project by breaking it into smaller, more manageable pieces.

1 Overall description

In the final tally, there are three animals that live in the zoo. You have some choice as to what the animals are in your version, and you'll get to pick what they look like, but for purposes of description we'll say they are a turtle, a lightning bug, and a dog. The turtle just basks on its rock and needs to be fed occasionally, but is otherwise sort of boring. The lightning bug flies all over but manages its own feeding. The dog is the most demanding, needing to be fed and also requiring personal attention, wagging its tail if it's happy enough.

Your immediate interactions with the system will be feeding the turtle, feeding the dog, and petting the dog.

Without any intervention from you, the lightning bug flies around and the dog wags its tail. If you go too long without feeding them, the dog and the turtle will die; if you just avoid petting the dog it will leave the zoo to go find someone who will give it attention.

Visual rendering

Each of the animals will have at least one image associated with it. The turtle and bug won't change, but the dog's tail will wag (more on this below). In addition to the main image, there will be a status bar below the turtle that grows when it's fed and shrinks as it gets hungrier; and two status bars below the dog, one for hunger and the other for happiness.

The actual images you use are up to you. If you have pictures of pets of your own, you can use those, or else you can draw your own or find something appropriate online.

2 Individual pet details

2.1 The sedentary pet

In the general description I called this one a turtle, but feel free to substitute another kind of basically sedentary pet that just requires food. This one doesn't have to move at all, but needs to be fed occasionally. It is fed by pressing the 'z' key, and one feeding will fill it up completely; it doesn't need to be fed again for 300 time ticks. If it isn't fed in that time, it dies, and no amount of feeding will revive it.

2.2 The mobile pet

This was a lightning bug, but you can substitute anything that flies around a lot and would feed itself. It moves back and forth across the window, and every time it hits a wall, it 'bounces' off the wall and heads in the other direction.

2.3 The needy pet

This was a dog, but you can substitute a different one (cats also work well in this role; they twitch their tails rather than wagging them, but the principle's the same). Pressing the 'm' key feeds it a serving of food, where each serving keeps the pet alive for 20 time ticks and the pet is full when it's got ten servings in its stomach. When it runs out of food (its stomach is empty), it dies, and no amount of feeding will revive it. Pressing the 'n' key pets it and makes it happy; if you have pet the dog in the last 100 time ticks, it is happy and wagging its tail, and if it's been longer than that it is unhappy and not wagging its tail. If it is unhappy for more than 100 time ticks, it leaves the zoo (disappears), and you can't pet it or feed it to bring it back.

Note that drawing tail wagging can be as simple as alternating between two different drawings every other time tick, although you can get fancier if you like.

3 Checkpoints

A significant part of the work on this project lies simply in the design—how to arrange the data and how to describe the functions you'll need. Furthermore, the later parts (actually writing Racket code) will go a *lot* smoother if your design is both complete and correct.

To that end, during the two intervening lab periods, the 18/19th and the 25/26th, you'll need to show me specific progress you've made on the project, as outlined below. Bring your design work with you to lab. It doesn't have to be particularly pretty, because you'll be explaining it to me (and I'll help you refine it).

Projects are graded out of 50 points, and each checkpoint is worth 10 of those points. So don't blow them off.

Checkpoint 1: Pet design

To begin, we'll pretend that each pet lives in its own completely separate world, independent of the other pets; this will let us focus on each piece separately. Fill out appropriate data worksheets for each kind of pet, and bring them with you to lab on the 18th or 19th. Make sure to include specific examples of states the pet might be in. You *don't* need any images at this point, just the basic status data, the information you'd have to write down if you wanted to stop the game and start it later right where you left off.

Checkpoint 2: Overall design, completed pet

By the following week, you should have at *least* the following completed (and bring it to lab):

- A data worksheet for the entire zoo (all three animals)
- For each separate pet, test cases for its tick and key functions
- For at least one of the pets, a basically complete implementation

The test cases and the implemented pet should be in a Racket file that you can demonstrate for me during the lab. This can be on the lab machine (in which case you should make sure the Racket file is in your network account

or on a thumb drive), but if you have your own laptop you're working on, you can demo on that instead.

Final version

Your final handin is due by the start of class on the 1st of November. If you need an extension for some reasonable reason, you need to talk to me *IN ADVANCE*.

4 Extras

I'll give a small amount of extra credit for extensions to the system, such as those below. Don't spend time on these unless you're pretty solid on the other parts; I can't give the extra credit if the main system doesn't work. If you'd like to try one of these but aren't sure about representation or techniques, come talk to me and I'll point you in the right direction.

- Animate the wagging tail as more than just switching between two images: show the tail in several intermediate positions as well.
- Move the lightning bug diagonally; hitting a wall will cause it to 'bounce' at an angle (kind of like a billiard ball).
- Make the lightning bug turn its light off and on at random. The Racket expression `(random 100)` returns a random integer from 0 to 99 inclusive.
- When the spacebar is pressed, the dog jumps towards the lightning bug, wherever it currently is. (This one's tricky!)