1 Administrative Topics

• We take the quiz

• Bruce and I have both tried to debug saveCanvas, but it doesn’t always work on all computers.

2 Going over Homework

When do we include num_bones as a parameter to __init__ and when do we exclude it? It depends on the design. When we design the class, we decide who should have control over the initial number of bones:

• The user/caller/main code must determine it. Then we include it as a parameter, and force the user to supply a value. We use that value to initialize the field.

```python
def __init__(self, name, num_bones):
    self.name = name
    self.num_bones = num_bones
```

• The user has the option of supplying the value. We include it as a parameter, but provide a default value, so the user can include or exclude a num_bones argument.
```python
def __init__( self, name, num_bones=0 ):
    self.name = name
    self.num_bones = num_bones
```

• We mandate the initial number of bones is some specific value (e.g. 0). We exclude it as a parameter, so the user has no ability to set the value.

```python
def __init__( self, name ):
    self.name = name
    self.num_bones = 0
```

### 3 Proj 8

In lab this week, you rewrote your L-system and interpreter code as classes. Why? This is part of a paradigm shift from top-down design using some object to an (almost) entirely object-oriented design (more on that on Mon). But there are two advantages that you should be able to see now:

1. Classes allow us to explicitly group data with the operations we would like to perform on that data. In other words, we can place the L-system contents with L-system functions together in a very clean manner. Also, this enables us to set up a very clear way of interacting with L-systems. We define L-system methods. The user must interact with an L-system using its methods. The user does not need to know anything about how an L-system is stored. Also, this is consistent with the emphasis I have placed on “protecting” data.

2. Classes allow us to provide names to different aspects of the data. Last week, we were using lists to store L-system data. We needed to remember that the first item of the list contained the base and the second item contained the list of rules. In the new version, we store the base in a data field named “base” and the rules in a list named “rules”.

Today, our goal is to write a main program for the project. Let’s read the instructions...
1. Write the goto, place, and orient methods in the Interpreter. This allows us to control where we will draw without including turtle in the main program.

2. Write the \_str\_ method for the Lsystem. This allows us to easily turn the contents of the Lsystem into a nicely formatted string.

3. Write a main program that draws at least two multi-rule L-system pictures with the contents (as text) of the L-system labelling them. How do we write a main program? We need to use objects. How many interpreter objects do we need? One. How many Lsystem objects do we need? One for each L-system.

Here is some code to draw two trees (without adding text):

```python
terp = interpreter.Interpreter(600,600)

# Tree 1
lsys = lsystem.Lsystem( 'systemD' )
lstr = lsys.buildString( 4 )
terp.place( -150, -100, 90 )
terp.drawString( lstr, 5, 25, 'tan' )

# Tree 2
lsys = lsystem.Lsystem( 'systemB' )
lstr = lsys.buildString( 4 )
terp.place( 100, -100, 90 )
terp.drawString( lstr, 5, 15.0, color='seaGreen' )

terp.hold()
```

### 3.1 Adding Text

Remember Bruce’s edict that we can’t draw anything without calling `drawString`. But what if we want to do something complicated, like add text? Here is a strategy:

- Add a case to `drawString` which handles a new symbol (e.g. T). This code should call the turtle code to add text to the picture (look it up on python.org).
• Add a parameter to `drawString` that enables the user to pass in the text that needs to be added. Give it a default value, so the user doesn’t need to supply a text string every time `drawString` is called (this would be annoying if the user was drawing a tree without any text). If you want to control the font size, you could add a font parameter as well, or you could just assume the distance parameter is a proxy for the font size. After all, these are analogous ideas.

• When you call `drawString`, pass it the string `T` as the string to draw, and pass it a value for the `text` parameter.