1 Administrative Topics

• We take Quiz 6

2 Adding Branching

I want to draw a plant, with branches. Then I draw up one branch. I want to draw the next. I need to get the turtle back to the base of the branch. To do that I need to implement a “memory”.

We add two new characters to the alphabet:

[ is save the turtle state (i.e. position and orientation)

] is restore the turtle state (i.e. position and orientation). In particular, restore the last state that was saved.

With the string “F[+F]F”, we draw forward, save our state, turn left 25 degrees, draw that branch by going forward, then return to the base of that branch (and our original heading) and draw forward again (see Figure 1).

We need to add support for the interpretation of the state-saving. We do this by saving the position and heading in a “stack”. A stack is a list, but with special rules – the last item added is always the first item removed.
Let’s look at a longer plant string: “F+[F+[F]F]F”. By storing the positions in a stack, we return to the most recent branch point every time we encounter a "]”’. The picture is shown in Figure 2.

To implement the turtle save/restore commands, we use a Python list as our stack. We insert an object onto the stack using list.append() and remove an object from the stack using list.pop()
The updated code for drawString is shown here:

```python
def drawString(distance, angle, string):
    stack = []
    for char in string:
        if char == 'F':
            turtle.forward(distance)
        elif char == '-':
            turtle.right(angle)
        elif char == '+':
            turtle.left(angle)
        elif char == '[':
            stack.append(turtle.position())
            stack.append(turtle.heading())
        elif char == ']':
            turtle.up()
            turtle.setheading(stack.pop())
            turtle.goto(stack.pop())
            turtle.down()
```

We test the new code with the above examples (using an angle of 25 degrees) (again, see Figures 1 and 2).

3 Advice for Project 7

- To add leaves or berries to trees, add a symbol (e.g. L for leaf and B for berry) to the L-system. I put my leaves just before each ] because I assume that is when the branch is ending. But you can play around with leaf placement.

- Then, add support for each new symbol in drawString. This code can be simple or complex, but obey this rule: LEAVE THE TURTLE AS YOU FOUND IT! I mean this: if you change the line thickness or color, then record the line thickness or color in a variable first. Then do your drawing. Then, set the values back.

- I added place to interpreter.py. You could add position/orientation parameters to drawString and make it call place at the beginning (just like block in project 3)
Here is an example of code to draw a leaf. It uses a different color and pen width than the branches use, so it saves and restores them:

```python
def draw_leaf(c, distance):
    if c == 'L':
        (pclr, fclr) = turtle.color()
        w = turtle.width()
        turtle.color('green')
        turtle.width(4)
        turtle.left(90)
        turtle.forward(distance)
        turtle.width(w)
        turtle.color(pclr, fclr)
```