Comp 1421 Foundations of Computing

Workshop2 Introduction to the WeeBee engine

CBPrice September 2020

Color Coding for this Worksheet	
	Information
	Instructed Learning
	Autonomous Learning
	Link to Assignments

Useful methods Full list on the module website

Moving To	
flyTo(X,Y);	moves in a straight line to (X,Y)
leapTo(X,Y);	moves in a parabola to (X,Y)
walkTo(X);	moves in a straight line to X
runTo(X);	moves with a wobble to X

Moving At	
jump(height);	jumps vertically with a jump height
spin(speed);	spins around centre with a speed
rest();	pauses USEFUL FOR SYNCHRONIZATION

Verbal, mental and emotive	
says("text");	displays text on screen preceded by actor/prop name
shouts("text");	displays text on screen without name
feels(emotion);	changes characters appearance

By the end of these activities, you should be able to answer mock exam question 2.

Activities

1 **Exploring the Engine**

(a) Download and unzip the engine. Then open up **activity1.cde** : File > Open, then select.

(b) Type this line of code **add(bigtree,70,10);** then hit <enter> then hit the run button

(c) Now add more scenery to make a pleasant scene. You can add as many instances of a particular piece of scenery as you like.

(d) Now choose **one** Actor. I shall choose Pip. Add your Actor somewhere like this **add(pip,40,10);**

(e) Now get your actor to jump, like this **pip.jump(30);**

(e) Now get your actor to move somewhere interesting, like this **pip.flyTo(60,40);**

(f) Now explore the other methods available. Investigate the ones given above, then perhaps look at the Code Method Summary on the website.

(g) Save your work: File > Save

2 **Working with 2 actors – synchronization**

Here you are going to practise coding two characters, to get their behaviour synchronised. This has been explained in class. There are two types of behaviour: *concurrent* where the characters move at the same time; and *sequential* where one character moves, and then the second one moves. Sequential and Concurrent processes are very important in Computer Science.

(a) Open **activity2.cde** then choose two characters and add them to the scene. I shall choose Pip and Flup.

(b) Get one to jump, and the other to rest:

```
pip.jump();
flup.rest();
```

(c) Now reverse the sequence

```
pip.rest();
flup.jump();
```

(d) Now get them both to fly at the same time, to the same place in the scene.

(e) Now get them both to stay there for a while.

(f) Now get one to fly out of the scene to the right, while the other jumps

(g) Finally get the other character to exit the scene to the left.

3 **Loops (1)**

Loops are one of three programming "constructs". The technical term is "iteration". Remember that word. There are many ways of coding iteration, but in reality, we only need one. Here we shall study the "while loop".

(a) Open up **activity3.cde** and *read the code*. In particular look for (i) where the loop counter is declared, (ii) find out what **type** of variable it is, and (iii) find out where and how it is updated, (iv) work out how many times you expect the loop to loop.

(b) Add a line of code to make the character jump 4 times. Set the jump height to 10. Run your code and count the jumps. Make sure she jumps 4 times.

(c) Change the code to make the character jump 7 times. Run and count.

(d) Now change the code to make the character jump higher and higher. **HINT:** you should use the value of **count** as a *parameter* of the **jump(parameter);** method.

(e) Now change the code to make Pip and Flup jump together for the same number of times.

4. **Loops (Challenging Mission)**

Your mission, if you choose to accept it is to make Pip jump 4 times, **THEN** Flup jump 5 times.

(a) Open up **activity4.cde** where you are given some template code.

(b) Now code the mission! **HINT:** You will need to use two while-loops.

5. **Tracing Loops**

Grab the Loop Trace Diagram from the module website and use this to manually trace through a loop.

6. **The "Notional Machine" [Optional]**

This is an optional activity. It is intended to help you understand the *Notional Machine* behind the WeeBee engine. That means understanding *what the computer is doing when it runs your code*.

As explained in class, when your code is executed, it first builds up an *execution list* for all the characters you have coded. When this is done, the engine processes the list for each character at the same time.

The table below shows some code (from activity2), on the left and on the right is part of the output from the log file which shows the execution list for Pip, Flup and the scene manager (SMan). The leftmost column shows the time when things happen.

So the first line of code, pip.jump(), puts the command JumpT onto Pip's execution list, and the second line of code, flup.rest() puts RestT onto Flup's execution list. So when the engine runs, Pip and Flup execute these commands

Your task here is to look at the code and the execution lists, and make sure you understand how they are related. You might wish to write a short paragraph explaining this.

```
pip.jump();
flup.rest();

pip.rest();
flup.jump();

pip.flyto(40,40);
flup.flyto(40,40);

pip.rest();
flup.rest();

pip.flyto(100,20);
flup.jump();

pip.rest();
flup.flyto(-100,100);
```

t	SMan	Pip	Flup
0.0		JumpT	RestT
2.0		RestT	JumpT
4.0		FlytT	FlytT
6.0		RestT	RestT
8.0		FlytT	JumpT
10.0		RestT	FlytT