Preparing to Teach

Introduction

Experience is the best teacher. But even the best teacher needs guidance on how to get that experience, efficiently. A busy teacher does not have the luxury of the huge amount of time needed to learn something using a pure discovery-based approach. This document is aimed to guide, but not constrain.

The approach presented here actually models one mode of classroom deployment we have used with great success with children in years 1-6, but especially in years 3&4. We call this "guided bricolage". Bricolage means tinkering, where the learner just explores what the code does, but without guidance. Pure bricolage does not really work, hence the need for guidance. I guess I used this approach when I was teaching physics in the 80's.

When you type a line of code (as explained below), you must type it exactly (case-sensitive) and most lines will end with a semi-colon. It is best to type **enter** when the line is complete. This will set you up for the next line.

Guided Bricolage – Part 1

Guidance – 1

a) Open **aScene1.cde**

b) Add a big tree **add(bigtree,70,10);**

Bricolage – 1

Choose some items of scenery, perhaps 4 or 5 and build up a pleasant scene. You can add multiple items, e.g., several instances of **bigtree.**

Guidance – 2

a) Choose a character. I shall choose **Pip**

b) Now add your character to the scene **add(pip,30,10);**

c) Now make Pip jump **pip.jump();**

d) Then find out what this parameter does **pip.jump(50);**

Bricolage – 2

Now try some other methods, such as **pip.spin();** Just add each line of code under the previous one. A good one to try is **pip.flyto(50,10);** where you can choose the co-ordinates. Just experiment, try to discover what each method does.

Guidance – 3

a) Open up aScene2.cde

b) Add some scenery to make a different scene

c) Add a single character of your choice

Bricolage - 3

Make your character move around and behave in a meaningful way, so the animation communicates something about the character. Perhaps an explorer, or a villain, or a playful friend.

Guided Bricolage – Part 2

Here we shall work with two Actors, and we shall learn how to co-ordinate their actions. This is the hardest aspect of the engine you will need to master. But there is a simple rule:

> *If you have two actors in your scene, then you have to tell them both what to do at each stage.*
> *That means you must add two lines of code, one for each actor.*

Guidance – 4

a) Open up aScene3.cde and add some scenery

b) Choose two Actors and add them to the scene where you like. I will choose Pip and Flup

c) Let's make them first do something together so add these *two lines of code* (remember the rule)

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

d) Now, after this, let's make Pip do something while Flup stands still, perhaps she is watching Pip. So add these *two lines of code*

```
pip.flyto(50,30);
flup.rest();
```

Bricolage – 4

Think about how to get your two Actors to move around and make some other actions, so that the animation makes sense. Perhaps a game of chase, or two friends coming together, having an argument and then making up.

Changing Scenes

By now you should be confident (i) how to create a coherent scene containing scene objects, (ii) how to use a basic set of methods to make Actors move-to places and also move-at a place, and (iii) how to code two Actors to have coordinated actions. My experience is that children at this point like to change the background image. This supports their creativity and opens up more space for experimentation (bricolage).

The background images are contained in the **data** folder, and they all have .jpeg format, and they all have width = 900 pixels and height 600 pixels. To access these backgrounds, you simply use their filenames; here's two ways of doing it:

a) To change the background but keep all Actors and scenery use

setScene("imageName"); e.g. **setScene("NightLake");**

b) To change the background and remove all scenery use

changeScene("imageName"); e.g. **changeScene("NightLake");**

It is not possible to remove Actors when changing a scene. You have to make them invisible, then change the scene, then make them re-appear. Here's one way of doing it, I shall use the single Actor Flup.

a) **flup.hide();**

b) **setScene("imageName");**

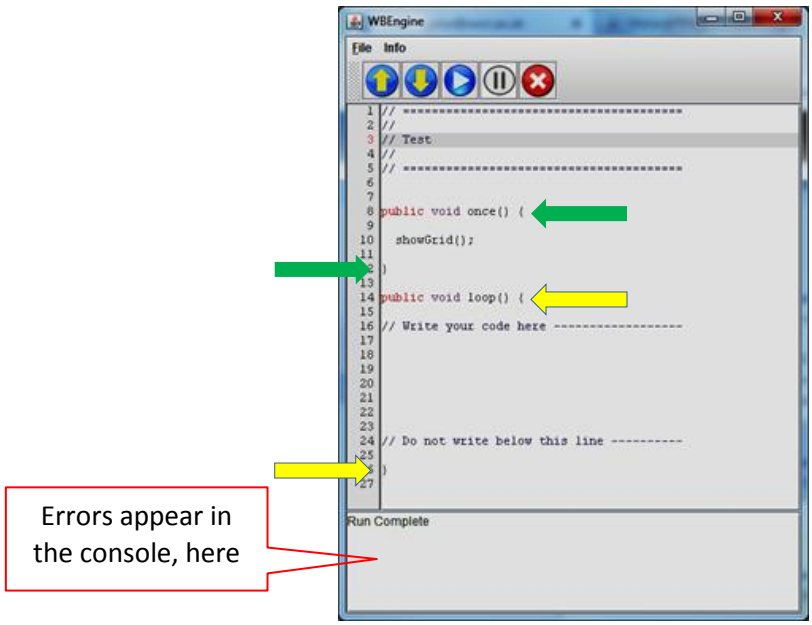
c) **flup.flyto(10,20);** This sets the location where you want Flup to be in the new scene

d) **flup.show();**

Errors you may encounter

Writing an engine to cope with all user errors is not easy, especially when you are not a professional programmer (like me). But the good news is children do not actually make many errors, in fact their error rate is no greater than my 'expert' undergraduates!

So what to do when there is an error? Errors are reported in the 'console' area indicated below. The first thing to check is that the two pairs of curly-brackets match up (green and yellow below), since these mark the start and end of two blocks of code. Some children manage to lose a bracket; why is a bit of a mystery, but seems to depend on the actual computer they are using. It rarely happens when children hit 'enter' after writing each line of code.



The engine will report an error by saying **Oops – there is a problem** in the console. Where possible, the line number is given.

So what errors can you expect and what should you look for?

1) Mis-typing of a word, e.g. **flap.jump();** instead of **flup.jump();** for **flup.jimp();** instead of **flup.jump();**

2) Forgetting to end a line with a semi-colon **;**

3) There may be problems with the **add** which should read **add(pip,10,30);** You may get some of the following which are incorrect: **add.barrel(30,10); add.(pip,10,30); add(pip,1030);** and variations on this theme.

X) There are some mistakes children make which are not strictly speaking errors. Here is the most common. They will correctly type **pip.flyto(20,10);** but they may incorrectly type **pip.walkto(20,10);** This should be **pip.walkto(20);** The engine interprets the second number as a time to complete the walk in seconds. This will make the animation run very slowly; if you see this, then look for this error.

Y) When the engine reports it is "Running" in the console for a very long time, and does not report "Run Complete", then it is hanging. Do a **File, Save** then exit the engine (red cross), start it up, load the file and look for a possible error before running again.