Programming with Alice

Program Design
Programming with Alice

Program Design

So far with Alice, we have learned:

- How to set up our world / scene
- How to create OBJECTS from CLASSES
- How to use PRIMITIVE and CUSTOM METHODS to animate our OBJECTS
- How to create our own METHODS to control the movement of our objects

We are now at a point in which our programs are becoming pretty complex

- Which means the problems we can tackle are more complex
We should now be spending much more time designing our program before trying to put it into the computer.

Good PROGRAM DESIGN involves thinking about the problem before writing any code. This allows us to identify (and refine) the steps of the solution without worrying about the programming language.

The design process is generally ITERATIVE. Meaning: the design can be reconsidered and modified multiple times until you get it just right.
Example in Alice – Create a Flying Bird
  – Seems easy enough…
  – flap (v1)
    - Flap the wings of the Bird

  Move the Bird Forward

…but unless your bird comes with a CUSTOM METHOD called “flap” these instructions won’t due much good
Example in Alice – Create a Flying Bird

- Let’s simplify what it is we mean by “Flap”:
  - flap (v2)
    - Both wings move up
    - Both wings move down
    - The Bird moves Forward

- But, we can’t just tell both wings to move up, so we make it:
  - flap (v3)
    - Left Wing moves up
    - Right Wing moves up
    - Left Wing moves down
    - Right Wing moves down
    - Bird moves Forward
Example in Alice – Create a Flying Bird

- This is getting close, but we’re missing the idea of timing. Let’s improve our program:

  flap (v4)
  - Left Wing moves up
  - Right Wing moves up  AT THE SAME TIME

  Left Wing moves down
  Right Wing moves down  AT THE SAME TIME

  Bird moves Forward
Example in Alice – Create a Flying Bird
– Or, even better:
  Flap (v5)
  ▪ Left Wing moves up
  ▪ Right Wing moves up
  AT THE SAME TIME

  ▪ Left Wing moves down
  ▪ Right Wing moves down
  AT THE SAME TIME

  ▪ Left Wing moves back to the middle
  ▪ Right Wing moves back to the middle
  AT THE SAME TIME

  ▪ Bird moves Forward
  OVER THE ENTIRE TIME
When the algorithm is written out as a well-thought series of steps, it is sometimes called PSEUDOCODE.

- It is written in easy to understand language, but is written very similar to the way that you would code it into your 3rd Generation Language.
  - Such as Alice, C++, C#, Java, etc.

- All it takes is a couple of changes, depending on the language you are using.
  - Your Pseudocode should be easy to translate into ANY programming language.
Example in Alice – Create a Flying Bird

- Alternatively, as a Flow Chart

```
start

Left Wing moves up
Right Wing moves up
Bird Moves Forward

Left Wing moves down
Right Wing moves down

Left Wing moves to middle
Right Wing moves to middle

Bird Moves Forward

end
```
Example in Alice – Create a Flying Bird

<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>pterodactyl.wingflap</td>
<td>No parameters</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>pterodactyl.rightWing</td>
<td>roll left</td>
<td>0.12 revolutions</td>
</tr>
<tr>
<td>pterodactyl.leftWing</td>
<td>roll right</td>
<td>0.12 revolutions</td>
</tr>
<tr>
<td>pterodactyl.rightWing</td>
<td>roll right</td>
<td>0.25 revolutions</td>
</tr>
<tr>
<td>pterodactyl.leftWing</td>
<td>roll left</td>
<td>0.25 revolutions</td>
</tr>
<tr>
<td>pterodactyl.rightWing</td>
<td>roll left</td>
<td>0.12 revolutions</td>
</tr>
<tr>
<td>pterodactyl.leftWing</td>
<td>roll right</td>
<td>0.12 revolutions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>pterodactyl</td>
<td>move forward</td>
<td>1 meter</td>
</tr>
<tr>
<td></td>
<td>duration</td>
<td>3 seconds</td>
</tr>
</tbody>
</table>