Visual programming languages in the Digital Technologies curriculum (F-6)

Darrel Branson - VCAA Digital Coding Specialist Teacher
In this session

• Where is visual programming in the curriculum?
• Some key programming concepts
• Introduction to visual programming
  – Solving a problem
    – Applying the problem solving methodology (PSM)
• Resources
Where is visual programming in the curriculum?
The Strands

Digital Systems
- hardware
- networks

Data and Information
- integrity
- representing data
- projects

Creating Digital Solutions
- analysing
- designing
- developing
- evaluating

Image credit: Paula Christophersen
Creating Digital Solutions

Explores processes and skills by which students create digital solutions

Four stages:
- Analysing
- Designing
- Developing
- Evaluating

Creating Digital Solutions requires:
- skills in using digital systems
- different ways of thinking (computational, design and systems thinking)
- interacting safely by using appropriate technical and social protocols.
## Creating Digital Solutions Levels F-6

<table>
<thead>
<tr>
<th>Levels F-2</th>
<th>Levels 3 and 4</th>
<th>Levels 5 and 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Define simple problems</td>
<td>Define problems in terms of data and functional requirements, drawing on previously solved problems to identify similarities</td>
<td>Design a user interface for a digital system, generating and considering alternative design ideas</td>
</tr>
<tr>
<td>Follow, describe and represent a sequence of steps and decisions (algorithms) needed to solve simple problems</td>
<td>Describe and follow a sequence of steps and decisions involving branching and user input (algorithms) needed to solve them</td>
<td>Design, modify and follow simple algorithms represented diagrammatically and in English, involving sequences of steps, branching, and iteration</td>
</tr>
<tr>
<td>Develop simple solutions as visual programs</td>
<td>Develop digital solutions as simple visual programs</td>
<td></td>
</tr>
<tr>
<td>Explore how people safely use common information systems to meet information, communication and recreation needs</td>
<td>Explain how student-developed solutions and existing information systems meet common personal, school or community needs</td>
<td>Explain how student-developed solutions and existing information systems meet current and future community and sustainability needs</td>
</tr>
</tbody>
</table>
Some key programming concepts
What are …

• visual programs?
• algorithms?
• control structures?
Visual programming

A programming language or environment where the program is represented and created visually rather than as text.

Examples of visual programming languages include: Alice, GameMaker, Kodu, Lego Mindstorms, MIT App Inventor, Scratch (Build Your Own Blocks and Snap).
Visual programming

[Image of Scratch and Sphero with blocks programming]

https://scratch.mit.edu/

https://edu.sphero.com
Algorithms

A description of the steps and decisions required to solve a problem.

Flowcharts are often useful in visualising an algorithm.

We use them everyday!

- Procedural text
- Solving a mathematical equation
- Recipes
- Our everyday routine
Control structures are the way a computer works its way through the coded instructions. All problems can be solved using control structures.
Sequence

Following step-by-step instructions, sequentially.

E.g. A recipe

Method

1. Combine flour, yeast and sugar in a large bowl. Stir it.
2. Use a wooden spoon to stir the mixture until well combined. ... hands to bring the dough together in the bowl. ...
3. Brush a large bowl with olive oil to grease. ...
4. Punch down the centre of the dough with your fist.
Branching occurs when an algorithm makes a choice to do one of two or more actions depending on sets of conditions and the data provided.

Making a decision!
If this is true, then do that, otherwise do something else
Branching

Diagram illustrating branching

Visual code blocks may look similar to these:
Iteration

Repetition of a process or set of instructions in computer programming.

Doing something over and over again!

Visual code blocks may look similar to these:
Visual programming
Programming stages

1. Define a Problem

2. Apply Problem Solving Methodology
   a. Analyse problem
      (Decompose, Abstraction)
   b. Design solution/s
      (Algorithm, Interface)
   c. Develop solution/s
      (Implement in Programming Language)
   d. Evaluate solution/s
      (Does it meet the needs of the problem?)
Thinking about what to program
(Defining a problem)
Defining a problem

Something that needs to be solved.

May need to…

...break it down into smaller parts (decomposition)

...identify the relevant parts (abstraction)

An Example:
Draw a regular hexagon with sides that are 50 steps long.
What is the problem asking us to do?  
(Analyse the problem)
Analyze the problem

What do we know about hexagons?
What are the specifics of our problem?

- How many sides does a hexagon have?
- How many angles are there in a hexagon?
- What is the angle of adjoining sides?
- How long are the sides?
- Where will I need to start drawing from?

SIX, equal length
SIX, equal angles
60 degrees
50 steps
What might a solution look like?
(Design solutions to the problem)
Creating a design (Algorithm) – Levels 3 & 4

Draw a hexagon algorithm 1:

1. Begin
2. If user wants a hexagon
   - draw a 50 step line
   - turn right 60 degrees
   - draw a 50 step line
   - turn right 60 degrees
   - draw a 50 step line
   - turn right 60 degrees
   - draw a 50 step line
   - turn right 60 degrees
   - draw a 50 step line
   - turn right 60 degrees
   - draw a 50 step line
   - turn right 60 degrees
3. Stop

Consider:
Other Control structures?
- Repetition? How many times is the same thing performed?
- Efficiency?

Check it. Does it work?
Creating a design (Algorithm) – Levels 5 & 6

Draw a hexagon - algorithm 2:

1. begin
2. If user wants a hexagon
   repeat 6 times
   draw a 50 step line
   turn right 60 degrees
3. stop

Check it. Does it work?
Moving from algorithms to a program
(Developing solutions using a programming language)
Let’s revisit our first algorithm.

1. Begin
2. If user wants a square
   • draw a 50 step line
   • turn right 60 degrees
   • draw a 50 step line
   • turn right 60 degrees
   • draw a 50 step line
   • turn right 60 degrees
   • draw a 50 step line
   • turn right 60 degrees
   • draw a 50 step line
   • turn right 60 degrees
   • draw a 50 step line
   • turn right 60 degrees
3. Stop
Code in Scratch
Levels 3 & 4
Using Branching

Branching IF... THEN...

When clicked
ask How many sides? and wait
if answer = 6 then
    clear
    pen down
    move 50 steps
    turn 60 degrees
    move 50 steps
    turn 60 degrees
    move 50 steps
    turn 60 degrees
    move 50 steps
    turn 60 degrees

Let’s revisit our second algorithm.

1. begin
2. If user wants a hexagon
   repeat 6 times
   draw a 50 step line
   turn right 60 degrees
3. stop

How might this look as a visual program?
Code in Scratch Levels 5 & 6

Adding iteration and user input
Does our solution solve the problem?
(Evaluating the solution to ensure it meets the problem’s requirements)
Evaluating the solution

Considerations:
- Has the solution solved the problem?
- Does it work? (Every time!)
- Does it do what it is supposed to do?
- Does it meet ALL of the requirements specified in the problem?

If the evaluation process determines that the solution does not satisfy the problem, then it may be necessary to revisit stages of the problem solving methodology again. This could continue until an accurate solution is reached.
Taking it further!  
(Adding to our original problem)

What about some other shapes?

- Triangle
- Rectangle
- Circle
- Square

What might the algorithms look like?
Taking it even further. Any shape?

Given the number of sides, identify the shape by name and then draw it.

Might include code to:

- Get more input from user (eg. number of sides)
- Determine the name of the shape (outputs name to user)
- Get more input from user (eg. length of sides)
- Calculate angle between adjoining sides (based on number of sides)
- Draw the shape
Common visual programming languages
## Some visual programming languages

<table>
<thead>
<tr>
<th>Language</th>
<th>Mobile Device</th>
<th>Windows/Mac</th>
<th>Possible Level</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scratch</td>
<td>Pyonkee (iPad)</td>
<td>Web-based or standalone</td>
<td>Level 3+</td>
<td>Easily accessible for schools</td>
</tr>
<tr>
<td>Scratch Jnr</td>
<td>iPad, Android</td>
<td>Web-based or standalone</td>
<td>Level 1+</td>
<td>No branching/selection.</td>
</tr>
<tr>
<td>Hopscotch</td>
<td>iPad</td>
<td>Web-based or standalone</td>
<td>Level 3+</td>
<td></td>
</tr>
<tr>
<td>MIT App Inventor</td>
<td>Web-based</td>
<td>Web-based or standalone</td>
<td>Level 5+</td>
<td>Creates Apps for Android devices</td>
</tr>
<tr>
<td>Alice</td>
<td>Cross-platform</td>
<td>Web-based or standalone</td>
<td>Level 5+</td>
<td></td>
</tr>
<tr>
<td>Kodu</td>
<td>Windows</td>
<td>Web-based or standalone</td>
<td>Level 3+</td>
<td></td>
</tr>
<tr>
<td>Snap!</td>
<td>Cross-platform</td>
<td>Web-based or standalone</td>
<td>Level 5+</td>
<td>Similar to Scratch. For devices.</td>
</tr>
<tr>
<td>Tynker</td>
<td>iPad, Android</td>
<td>Web-based or standalone</td>
<td>Level 3+</td>
<td>Similar to scratch.</td>
</tr>
</tbody>
</table>
Extending visual programming to devices!
Block programming - Robotic devices

Sphero Edu App

Microbit.org

Ozoblockly.com
Visual programming languages

Some ‘Digital Devices’ that use block-based programming

- Sphero robot
- Edison robot
- Dash & Dot robot
- Ozobots
- mBot robot
- Micro:bit
- Hummingbird
Resources
Learning more about visual programming

• Code.org - https://studio.code.org
  – Start with ‘Classic Maze’

• Scratch - http://scratch.mit.edu
  – Introduction to Scratch tutorial and activity cards

• CoderDojo (Resources)
  - https://coderdojo.com/resources/
  – Beginner Scratch (dojo and sushi cards)