Library of controls and procedures

In this document, we list several commonly used controls and procedures, which can be imported by any Snap program. Each library is stored in a sprite. Use the "load sprite from file" option to import the library into your program. You can then copy-paste to other sprites the procedures you are interested in.

Download the library by clicking here.

Radio buttons:

This sprite allows you to select one of 3 speeds: slow, medium, fast

Each click sends out an appropriate message. For example, if you click on "slow" the sprite sends "speed slow" message.

This sprite works only if you use it as is, with NO change in orientation or size.

File: radio.xml

How to use the control:

1. Import the sprite into your program.
2. Place the sprite at a suitable place with NO change in its orientation or size.
3. Process the 3 messages "speed slow", "speed medium", and "speed fast" to set the desired parameter in your program.

Snap and CS Concepts Used

When we design this control, we made use of the following Snap and CS concepts.

- Algorithms
    - Designing new algorithms
- Arithmetic
    - Expressions
    - Basic operators (+, -, *, /)
    - Advanced operators (abs)
- Concurrency
    - Synchronization using broadcasting
- Conditional statements:
    - Conditions: YES/NO questions
    - Relational operators (=, <, >)
    - Conditionals (IF)
    - Conditionals (If-Else)
    - Conditionals (nested IF)
    - Boolean operators (and, or, not)
- Costumes

- Data types – basic
    - Integers
- Events
- Sequence
- Variables
    - properties

For advanced users:

If you wish to resize the control:

Presently, the distance between left (right) edge of the slow (fast) circle from the center of the med circle is 50. And the radius of all circles is 12. Notice these numbers in the code and change them according to how you resize.

If you wish to use the control for some other purpose (e.g. Level: high/med/low):

- Keep only the 1st costume and delete the rest.
- Change the text labels to whatever you want.
- Duplicate the costume to create 2 more. Name them 'med' and 'fast' respectively. Fill the circles as before (blue for med and purple for fast).
- Go to scripts and change broadcast messages.

Design:

We basically use the geometry to design this control. The sprite's center is also the center of the "med" circle. When the user clicks, we find out whether they clicked on any of the 3 radio buttons, but comparing the x and y values of the pointer with the sprite's center. Here is the algorithm:

```
Algorithm which radio button was clicked:
When sprite clicked:
Given:
Xm = X of mouse pointer
Ym = Y of mouse pointer
X, Y = coordinates of the sprite's center (also the center of "med")
R = radius of all 3 circles (identical)
D = how far the sprite center is from the outer edge of "slow" and
"fast" circles.

Procedure:
If Xm is within R of X AND Ym is within R of Y
      "Med" was clicked
```

```
Else:
      If Xm is within D of X AND Ym is within R of Y
            If Xm < X
                  "Slow" was clicked
            Else
                  "Fast" was clicked
            End if
      End if
End if
```

Geometric shapes – 1:

This library consists of procedures to draw "regular" shapes such as square, pentagon, circle, etc.

File: shapes-1.xml

Here is the complete list:

- Triangle(size)
- Square(size)
- Pentagon(size)
- Hexagon(size)
- Rectangle(width, height)
- Regular polygon(size, number of edges)
- Circle with diameter input
- Circle with circumference input
- Semi-circle with diameter input
- Quarter-circle with diameter input

How to use the library:

1. Import the sprite into your program.
2. Copy-paste procedures for the shapes you are interested in.

Snap and CS Concepts Used

When we designed this library, we made use of the following Snap and CS concepts.

- Algorithms
    - Designing new algorithms
- Arithmetic
    - Expressions
    - Basic operators (+, -, *, /)
- Data types – basic
    - Integers
- Events
- Looping – simple
- Pen art
- Sequence

- Procedures – with input

Design:

We use basic geometry and the TRT principle to design these procedures.

Geometric shapes – 2:

This library consists of procedures to draw the same "regular" shapes as in Shapes-1 above except that they are filled, i.e. painted.

File: shapes-2.xml

Here is the complete list:

- FTriangle(size)
- FSquare(size)
- FPentagon(size)
- FHexagon(size)
- FRectangle(width, height)
- FRegular polygon(size, number of edges)
- FCircle with diameter input
- FCircle with circumference input
- FSemi-circle with diameter input
- FQuarter-circle with diameter input

How to use the library:

1. Import the sprite into your program.
2. Copy-paste procedures for the shapes you are interested in.

Snap and CS Concepts Used

In addition to the concepts required for shapes-1 above, we need to know the following concepts:

- Fill

Design:

We use basic geometry and the TRT principle to design these procedures.

String routines

This is a collection of routines to manipulate strings.

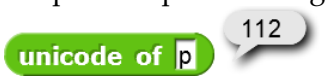
File: strings.xml

Here is the complete list:

- Uppercase(inStr): returns uppercase version of inStr, returns non-alphabets unchanged.

Design:

Snap offer operators to get and set the Unicode encoding of every character:



The encoding values of A thru Z and a thru z are contiguous. It so happens that the A-Z range (65 to 90) is lower than the a-z range (97 to 122). The gap between the two ranges is 32. We can use this information and these reporters to do the case conversion as follows:

```
Algorithm Uppercase:
Input: inStr (string of letters)
Output: outStr
outStr = initially empty
For every letter C in inStr:
      If C is outside the range a-z:
            Append C to outStr
      Else:
            Cval = Unicode value of C - 32
            Get equivalent letter of Cval and append it to outStr
      End if
End for
Return outStr
```

Author: Abhay B. Joshi (abjoshi@yahoo.com)
Last updated: 4 February 2021