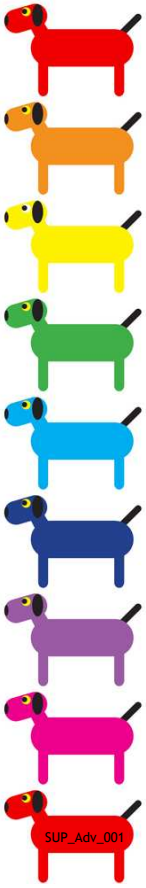


2Create a Superstory

Advanced User Guide

Contents

Contacting 2Simple

General Enquiries
Web: www.2simple.com
Email: info@2simple.com
Tel: (+44) (0)20 8203 1781
Fax: (+44) (0)20 8202 6370
Post: 2Simple Software, Enterprise House, 2 The Crest, Hendon, London, NW4 2HN

Useful Links
2CASS Home page www.2simple.com/super
AR viewer www.2simple.com/ar
2Simple Blog www.2simple.com > 2Simple Blog

External Sites
2CASS archive www.2cassarchive.co.uk
(2Simple does not maintain this website and is not responsible for its content)

Tech Support contact details
Phone: (+44) (0)20 8203 1781
Email: support@2simple.com
Web: www.2simple.com > support
www.2simple.com > support > Networks
www.2simple.com > support > technical questions > 2Create a Superstory
Faulty Media - should the CD-ROM develop a fault we will replace it free of charge.

System Requirements

Operating System: **Windows XP / Vista / Win7**
Screen resolution: **1024 x 600 or above**

CD-ROM Drive: Required for Installation
Printer: Required for AR screen
Soundcard & speakers: Required to view videos and sound elements within the program
Microphone: For recording voice-overs and lip-syncing.
Mouse, Keyboard: Required

Adobe Flash Player required
Adobe Reader required to view the user guide
Both of the above are available on CD-ROM but we recommend downloading and installing these from www.adobe.com as this will ensure you get the latest version.
Adobe Flash Player is not included within the main MSI installation—it is included as a separate install on the CD.

Note: This program was tested briefly on Windows 2000 and no issues were found. However, since we did not conduct extensive testing on this OS, it is not listed as supported.

Getting Started

This document is the Advanced User Guide for 2Create a Superstory.

Before you read further, please make sure you have already familiarised yourself with the main User Guide for this program which contains detailed information about most aspects of using the program. The main user guide can be accessed here: www.2simple.com > support > downloads.

The advanced user guide, which is detailed here, contains information about how to add ActionScript to your stories. This is an exciting and powerful facility, but it can be potentially quite complex as well. **It is only intended for experienced users who are comfortable with the standard use of the program and want to take it a step further.**

Using ActionScript involves adding your own *program code*. Many ActionScript code rules and conventions are not included in this guide; this guide is meant to provide a stepping stone to help the pupil understand the basics of coding.

Within the advanced user guide itself there are varying levels of difficulty. You can use ActionScript to code a simple horizontal movement for an object, or you can use it to code a swinging arm movement which uses a trigonometric function based on the amount of time elapsed.

Whatever level the code is used on, we hope it inspires creativity and stimulates your pupils!

Advanced — ActionScript

Like 2Simple's 2Do It Yourself, you can use Adobe ActionScript 2 to control many aspects of the program.

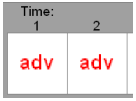
What you can control using ActionScript

Each picture area in the story, and each object within each picture area, has its own defined name and can be accessed and controlled using ActionScript.

Objects are any of the items in the right column on screen (dogs, cars, houses, etc). As you have seen previously, you can add your own animations to the picture and to objects by simply selecting an animation and dragging it to the timeline of the object or picture. With ActionScript, you have further flexibility in that you can create your own animations for objects or modify existing ones.

Places where you can add your own ActionScript

1. Start: You can add code that will execute once only, at the beginning of the story, by right-clicking the play button (the green triangle at the top of the page). Here, you can add your own predefined variables or methods which can be used later on in other code segments.



2. Animations: You can add code to the timelines of objects or pictures by dragging the "adv" action (near the bottom of the list) to the object or picture timeline. A code block in a timeline frame will execute repeatedly 30 times within 1 second when you press play.

3. Click events: You can add code to the click event of an object by dragging the "adv" action to the mouse icon, to the left of the timeline. When the object is clicked during play mode, the code block will execute 30 times and then stop.

Examples to get started

1. Drag any object to the picture. In the Edit Object window, drag the "adv" action to frame 1 and type "**this._x++;**" (excluding quotes) in the code block and click ok. Press play - the object will start moving to the right. Short explanation: “this” refers to the current object. “x” refers to the object’s horizontal position on screen. “++” tells the program to add 1 to the horizontal position each time this command is called, which will be 30 times per timeline frame. Always make sure to end each statement with a semi-colon.

2. Start a simple story and click on the Edit Picture Timeline button (the clock at the top of the screen). Drag the "adv" action to frame 1 and type "**pic._y++;**" (excluding quotes) in the code block and click ok. Press play - the entire picture will start moving down. Short explanation: “pic” refers to the current picture. “y” refers the the picture’s vertical position on screen.

3. Make giraffe’s neck grow longer - add the following code to a giraffe’s timeline or click event.
this.p3._xscale++;
this.p3._y--; this.p3._x-=.5;
this.p4._y--; this.p4._x-=.5;
Short explanation: p3 and p4 are the head and the neck of the giraffe. “xscale” is the extent to which the neck is stretched in the x direction, although this might not be horizontal if the object is rotated.

4. Add a “drag” activity object to screen and then any additional object as well. Add to the Picture Timeline: “**_root.pic.ob2._x = _root.pic.ob1._x;**” and then copy the Adv block to the rest of the timeline using shift+click. Press play: when you drag the activity object, the other object will partially imitate it by moving to the same x position.

Advanced — ActionScript

Items you can access within the different code blocks

1. Startup scripts

pages You cannot refer to individual objects or picture areas within the startup script, although you can refer to pages. You can use this area to define your own variables and functions which you can then refer to later on. To define the variable, simply type for example "var age:int=3;" (other variable types include Number and String). Remember to precede the variable name with "**_root.**" when referring to it later on (similarly for functions). An example function: "function move(ob:object):Void { ob._x++;};". Example usage, when used within an object animation code block: "_root.move(this);"

2. Object animations / click events

this Refers to the object to which the animation has been added. You can also use "_root.ob". You cannot access other objects here - only the object whose animation you are editing.

3. Picture animations

pic Refers to the currently selected picture area (there may be multiple picture areas on the screen, eg 2x2 story type). You can also use "_root.pic".

_root.pic.ob1, ob2... In the picture animation script you are able to access each object within the picture. Each object that you add to a picture is automatically assigned a unique name, starting from "ob1", then "ob2" etc. You can find out the name of an object by clicking on it once - the name will appear in light blue at the top right of the screen. In page 2 of the 2x2 story type, object names start from ob4.

Object Parts

Each object has a number of sub parts which you can access individually. For example, the car has 4 parts - an upper and lower section, and 2 wheels. This allows you to assign animations not just to the whole object, but to any of its parts.

The parts are named sequentially, starting from **p1,p2**... To find out the name of a particular part, look in the object's **layout.xml** file. Go to the location where the program was installed, typically C:\Program Files\2Simple Software\2CASS. Open the "objects" folder, and then the categories folder, eg "vehicles". Open the folder for the specific object, eg "car". Open the layout.xml file in a text editor such as Notepad. There are a number of shapes defined in the file—for example "rect" (rectangle), "oval", "roundrect" (rounded rectangle). Each shape has a set of dimensions defined for it as well as a unique number, referred to as "num=..." Based on the shape and dimensions you can work out which part has which number—eg for the car, the wheels can be accessed using "p2" and "p3".

So if I wanted just one of the car's wheels to rotate, I could add to the animation block for a car "this.p2._rotation++;"

In addition to the parts specified above, each object also has a speech bubble part which can be referred using "speech". Eg "this.speech._y = 100;" This will only work if you have added a "show speech bubble" action to the timeline before using the above code.

Properties of Picture Areas, Objects and Object Parts which can be changed

Properties should always be preceded with an underscore "_" when referred to in code.

Property	Explanation and possible values
_visible	true / false. Hidden objects do not respond to click events.
_alpha	0-100. This refers to the level of transparency. 0=transparent, 100=opaque. Values in between give varying degrees of transparency. Even fully transparent objects with alpha=0 can respond to click events.
_rotation	0 - 360. Values outside this range (above or below) are also valid, but are scaled back into this range to determine the degree of rotation, eg 365 = 5, -10 = 350. If you steadily add to the rotation it will cause the object to rotate clockwise. If you steadily subtract it will rotate anticlockwise.
_x, _y	Horizontal and vertical position. See next page for possible values.
_width, _height	Object width and height. See next page for possible values.
_xscale, _yscale	Resizes an object as a percentage of its original width or height. 100 indicates 100%, ie default scale. 200 will double the scale in the x or y direction. 50 will halve it. Note that objects are already scaled down by default when dragged onto canvas, so setting the scale to 100 will actually enlarge the object. The current scale is shown on screen at the top right in light blue when you click on the object's green circle. If the object has been rotated, the scaling direction will also rotate correspondingly. You can set this to a negative value; this will flip the image.

Different story types have different size picture areas - see the table. When you set the x, y, width and height values for an object, take note of the boundary values for the picture area which contains it.

NB: From the perspective of an object, x=0, y=0 represents the CENTRE of the picture area, not the top left. The x and y range for Simple screen is therefore (x: -380 to 380, y: -180 to 180), and to set an object to the top left in Simple screen we would use x=-380, y=-180. The reason this has been done is to enable screen rotations to work as expected.

Similarly, when you set the x and y position of an object this represents the CENTRE of that object. So if you set the x value of an object on the simple screen to –380, this will result in half the obect going off the picture boundary. To ensure the object stays within the boundary you could for example use "this._x = -380 + this._width/2;"

Screen	Width, Height	Width, Height - pg2,3... if diff from pg 1
Simple, Auto, Activity	760, 360	
Full screen	800, 600	
Square book	360, 250	360, 360
Landscape	260, 130	260, 180
Portrait	360, 330	360, 530
2x2	360, 330	160, 240
Textured	360, 330	
Scroll	680, 310	
Reference	320, 410	320, 510
Screen Banner	User chooses	

Disabling rotation during line follow

In the main user guide it was stated that you can disable rotation during line follow. They way you can do this would be to add a "this._rotation=0;" code block to each timeline frame that has a line follow action.

There are many predefined ActionScript functions and values you can use - see Adobe ActionScript Language Reference for a full list. Here are a few to give you some idea:

Math.random() - returns a psuedo-random number between 0 and 1.

Key.isDown() - takes an integer argument, referring to a specific key, and returns true if that key is currently being pressed, or false otherwise. Some key values: left=37,up=38,right=39,down=40. Example "if (Key.isDown(37)==true) { this._x--; }" will cause the object to move left if the left arrow key is being pressed.

Many object animations rotate object parts back and forth - these animations often make use of the following:
Math.PI - the value of the mathematical constant, PI
Math.cos(), Math.sin() - trigonometric functions. They take an angle as an argument (in radians) and return the cos or sin of that angle.

2CASS also defines some of its own functions and variables which you can use:
_root.moveToNextPoint(this) - line follow action
_root.resetPage(_root.curPage) - reset all page objects and variables to what they were at the start.
_root.showNextPage() - moves to next page (only for some templates such as simple and full screen)
_root.showPage(...) - moves to a specific page number (only for some templates such as simple and full screen)
_root.t - this variable stores the number of frames that have elapsed since the current page started (30 frames per second)

Pages
Each page in a story can be referred to using **_root.page1, _root.page2** etc. Books with pages which turn (Square book, Portrait, Landscape etc) have **root.page1R**, **_root.page2L** (right and left) etc. You can modify the page properties as in pg 10 above. Pages also contain turn page arrow objects, named **nextBut** and **prevBut.** You could for example make pg 1's next arrow invisible using "_root.page1.nextBut._visible=false;" This code can be used in the startup code block in addition to "adv" code blocks for objects or picture areas.

Advanced

Collision detection

You can add code which will execute when 2 objects overlap with each other. For example, add the following code to the picture area timeline: "

```
if (    (Math.abs(_root.pic.ob1._x - _root.pic.ob2._x) < 10 ) &&
        (Math.abs(_root.pic.ob1._y - _root.pic.ob2._y) < 10))
{ _root.pic.ob1.rotation++;}
```

This code checks if the x and y values of 2 objects are close together—and if they are, it rotates one of the objects. Maths.abs returns the absolute value of its argument.

Adding your own objects

Go to the location where the program was installed, typically C:\Program Files\2Simple Software\2CASS. Open the "objects" folder. Each subfolder within this is an object category. Each folder within a category is for an individual object. You can add your own categories or objects by simply adding folders to this structure, with the relevant folder content. Each object folder has a number of files:

- **layout.xml** - this file defines the locations and dimensions of each object part.
- **preview.bmp** - a picture thumbnail which the program will use to represent the object
- **animations** - a set of actionscript files (.as extension) and corresponding bmp files which serve as thumbnails for the animations.

The standard animations that are available to all objects are available separately in the "\resources\standard animations" and "\resources\move scripts" subfolders.

The code for the standard picture animations can be found in "\resources\pic animations".

If you edit or add to any of the actionscript files above, add the code within the curly braces "onClipEvent (enterFrame){ ...}".

Copyright Information

2Simple Software, United Kingdom
info@2simple.com
www.2simple.com

Userguide written by D Ziskind
Last updated 17 Feb 2010 9:08

2Create a Superstory

Thanks to everyone who helped make this product - *Max Wainewright, 2010*

2Simple Software
Enterprise House, 2 The Crest
Hendon, London NW4 2HN
Tel: 020 8203 1781 ~ Fax: 020 8202 6370
www.2simple.com ~ info@2simple.com