MIPS Syscall Code 100

(files: mars/mips/instructions/syscalls/SyscallGame.java and mars/MMIOInput.java inside the Mars .jar zipped file. Unzip the .jar. Run Windows script compileJavaGame.bat inside the .jar to compile the files and create the updated Mars .jar with them. SyscallGame.java in Mars…Tester…jar contains the code of a testing program for the Submarines game. It opens up when the game screen opens up.)

Overview

A MIPS syscall of code 100 is defined in our special Mars package. It is for the game development using a MIPS program running in Mars.

Game Framework

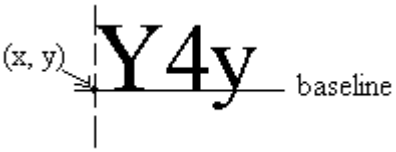
A game is composed of a number of game objects. It can also have an array of images stored in it. Each game object created has an unique ID for identification. A game object can represent one of the following three things (or none of them):

- Image – by associating the object with a non-negative index number to an image in the image array of the game.
- Text – by associating the object with a single-line text string. Note that new-line characters such as "\n" inside the string will be removed.
- Integer – by associating the object with an integer.

An object can represent only one piece of Image, Text or Integer. Setting an association of an object will remove any other associations of it. For example, associating Text with an object will remove the Image or Integer currently associated with it. To make an object represent nothing, we can associate it with a negative image index number. That is also the initial association of an object when it is created.

Game Drawing

The game draws each game object (if it does not represent nothing). The objects are drawn in an ascending order of their IDs. The game screen's pixels go from left to right in the x-axis and from top to bottom in the y-axis. Both x and y values start from 0. Each game object is drawn based on its pixel location (x, y), which can be set using the syscall. For drawing an Image object, the associated image will be drawn with its top-left corner at the (x, y) pixel of the game screen. For drawing Text or Integer object, the associated text string or integer will be drawn in such a way that the (x, y) pixel is the left-most pixel of the baseline of the text or integer. The following illustration shows where the baseline is.



If an object is Text or Integer, the color and font used for drawing it can also be set using the syscall.

Game Input

Any keystrokes on the game screen will be stored as the input on the MIPS program using Memory-Mapped Input Output (MMIO). Thus, the MIPS program should check or read the input according to the MMIO specification. In this game every keystroke generates an input. But the ASCII code stored in the input for a non-character keystroke is undefined. Also, the game does not support pressing or holding down multiple keys at the same time.

Syscall Usage

For this syscall of code 100, $v0 is set to 100. $a0 is the *action code* set to indicate what action the syscall should do. The parameters for an action will be passed using some other registers (see the table below). Any errors generated during the syscall's action will terminate the MIPS program immediately. But the game screen, if any, will not be closed for debugging purpose. It can always be closed by clicking its window's Close button (the 'x' icon on the top-right corner).

Action	**Action code ($a0)**	**Parameters**	**Example**
Create game	1	$a1 = game screen width. $a2 = game screen height. $a3 = base address of a string for game's title.	.data title .asciiz "Star Wars" .text li $v0, 100 li $a0, 1 li $a1, 800 li $a2, 600 la $a3, title syscall
Create game objects (any existing ones will be	2	$a1 = number of objects (at most 231) to create. *All the objects initially represent "nothing" and are*	li $v0, 100 li $a0, 2 li $a1, 20 syscall

removed first)		*assigned the IDs 0, 1, 2, …, $a1 – 1, respectively.*	

Set the image array in game	3	$a1 = base address of a string of the images' file paths (see the Note below) delimited by semi-colons. The images will be stored in an array in the game. *The order of the images in the string is preserved in the array with the array index no. starting from 0.*	.data images .asciiz "background.gif; tank.gif;gun.gif" .text li $v0, 100 li $a0, 3 li $a1, images syscall
Create and show the game screen	4		li $v0, 100 li $a0, 4 syscall
Set the game background image	5	$a1 = index no. to the background image in the game's image array, or a negative number for filling the background with a black color.	li $v0, 100 li $a0, 5 li $a1, 47 syscall
Redraw the game screen showing any updates of the game objects since the last call of Redraw	6		li $v0, 100 li $a0, 6 syscall
Close the game window and destroy the game	7		li $v0, 100 li $a0, 7 syscall
Set game object to represent Image	11	$a1 = object ID $a2 = index no. to an image in the game's image array, or a negative number for resetting this object to represent nothing.	li $v0, 100 li $a0, 11 li $a1, 17 li $a2, 0 syscall **or** li $v0, 100 li $a0, 11 li $a1, 17 li $a1, -1 syscall

Set game object's location	12	$a1 = object ID $a2 = x-coordinate $a3 = y-coordinate	li $v0, 100 li $a0, 12 li $a1, 4 li $a2, 220 li $a3, 342 syscall
Set game object to represent Text (single-line text only)	13	$a1 = object ID $a2 = base address of the text string. *Any new-line characters such as "\n" inside the string will be removed.*	.data scoreText .asciiz "Score: " .text li $v0, 100 li $a0, 13 li $a1, 56 la $a2, scoreText syscall
Set game object to represent Integer	14	$a1 = object ID $a2 = the integer	li $v0, 100 li $a0, 14 li $a1, 56 li $a2, 970 # i.e. a score syscall
Set game object's color for drawing Text or Integer	15	$a1 = object ID $a2 = integer for a color that is a combination of red, green and blue components (byte 0 for blue, byte 1 for green, byte 2 for red, byte 3 is not used). *An object has a black color set initially.*	li $v0, 100 li $a0, 15 li $a1, 56 li $a2, 0xff00ff # purple syscall
Set game object's font for drawing Text or Integer	16	$a1 = object ID $a2 = font size $a3 = non-zero for using Bold font, or else 0. $t0 = non-zero for using Italic font, or else 0. *An object has a font of size 16 and Plain style set initially.*	li $v0, 100 li $a0, 16 li $a1, 56 li $a2, 32 li $a3, 1 li $t0, 0 syscall
Play sound in loop	17	$a1 = sound ID	li $v0, 100 li $a0, 17 li $a1, 0

			syscall
Play sound for once	18	$a1 = sound ID	li $v0, 100 li $a0, 18 li $a1, 0 syscall

Note: Use only GIF, PNG or JPG images. The file path of an image can be specified using *Relative* path (e.g., "back.gif" or "img/car.jpg") or *Absolute* path (e.g., "c:/img/car.jpg" on Windows or "/img/tank.png" on Unix). A Relative path is relative to the current user folder of running Mars (by default, the folder of the Mars .jar program file). "/" not "\" should always be used to separate folders in specifying the path, even when running Mars on Windows.