Enable data reporting

Disable data reporting

Enable stream mode

Project Status

Devices and functions of each device:

Timer:

The timer is used to move both players even if it’s not their intention, since the difficulty factor of this game is that the players can’t be standing still.

In this game, we are using the timer with interrupts, therefore we had to implement the functions:

* timer\_subscribe\_int (): used to subscribe the timer 0 interrupts.
* timer\_unsubscribe\_int (): used to unsubscribe the timer 0 interrupts.
* timer\_int\_handler (): used to handle the interrupts from timer 0 (each time the timer 0 raises an interrupt, the interrupt handler will increment a global variable (totalInterrupts)). Every time totalInterrupts is a multiple of 20 (multi computer mode) or 5 (single computer mode), the main loop calls the function that move the players (passive\_move\_payers()).

Keyboard:

The keyboard is used to move the front-end of each player’s light trail every time the player desires.

In this game we are using the keyboard with interrupts, therefore we implemented the following functions:

* keyboard\_subscribe\_int (): used to subscribe the keyboard interrupts, in excusive mode.
* keyboard\_unsubscribe\_int (): used to unsubscribe the keyboard interrupts.
* kbc\_ih (): used to get the key code associated with the keyboard interrupt.

Video card:

The video card was used to display the game to the user. We chose to run the game in mode 0x115, owing to the fact that this mode has 24 bits per pixel (1 byte per colour) so the range of colours available is wide. Furthermore, since we are using the function xpm\_load () provided in the labs and because this function was programmed to handle xpm that only have 1 or 3 bytes per pixel, the mentioned mode is compatible with xpm images that have 3 bytes per pixel.

To use the video card properly, we use the following functions:

* new\_vg\_init (): uses function vbe\_get\_mode\_info () provided in the lectures to get the information about a specific graphic mode. After that, the function maps the Video RAM in the process’s address space, so that our game can change what is displayed on the screen. Finally, this function calls another function called set\_mode () that sets the graphic mode of the video card.
* set\_mode (): programs the video card to operate in the mode specified.
* vg\_draw\_pixel (): colors the pixel in the specified coordinates with a given color.
* vg\_draw\_rectangle (): draws a rectangle in the specified position filled with a given color.
* vg\_draw\_hline (): draws a horizontal line with a given length starting in the specified position and coloured with a given color.
* The collision detection is implemented by function find\_color () that checks if a certain position is occupied or not (if the color in that position is black, then there is no collision).

Mouse:

The mouse is used to choose the desired option in the menus.

Since we are using the mouse with interrupts, we had to implement the following functions:

* mouse\_subscribe\_int (): used to subscribe the mouse’s interrupts in exclusive mode.
* mouse\_unsubscribe\_int (): used to unsubscribe the mouse’s interrupts.
* mouse\_ih (): used to get the byte associated with each interrupt.
* parse\_mouse\_info (): used to convert the raw bytes received from the mouse into a more manageable data structure (struct packet provided in the lab).

Serial Port:

The serial port was used to create a game mode in which the two players are playing in different computers. The serial port is programmed to operate with half-duplex communication, with a bit rate of BITRATE. Furthermore, we make use of FIFO’s in both transmitting and receiving data. Finally, we use pooling to send data to the other device and interrupts to receive. Regarding the data sent, each time a movement key is pressed, the direction of the movement (an integer which can be any number between 0 and 4, inclusive) is sent to the other computer.

We chose not to use the centralized approach, in order to minimize the latency in the movements of the players. Therefore, each computer has almost the same code and sends the input received to the other one, so that the processing of each movement is done by both computers. However, this requires a more refined synchronization between the computers, so in order to address that problem, when one user chooses to play the multi computer mode, the computer sends a specific character to the other one and waits for a response (function wait\_for\_connection ()). When the response arrives, both computers start the game.

FALAR DO PROTOCOLO

Falar das funcoes

Real time clock:

This device was used to enable and disable night mode. NAO SEI MAIS