Simulating mouse clicks by writing to memory

Conquest of the Longbow SCI Version 1.000.168

The game uses two pointers `stack_ptr0`, `stack_ptr1` found at offsets `0x1C8AC` and `0x1C8AE` (relative to a base pointer which needs to be determined separately) that are used as signals to the game engine that new input is available and should be processed.

	Address offset
stack_ptr0	0x1C8AC
stack_ptr1	0x1C8AE

These point to a 16-length input queue that contain information about keyboard and mouse events. Each event is described by 14 bytes long making the size of the input queue 16x14 bytes.

	Address offset	**Pointer value (decimal)**
event-data0	0x1D71A	14666
event-data1	0x1D728	14680
event-data2	0x1D736	14694
event-data3	0x1D744	14708
event-data4	0x1D752	14722
event-data5	0x1D760	14736
event-data6	0x1D76E	14750
event-data7	0x1D77C	14764
event-data8	0x1D78A	14778
event-data9	0x1D798	14792
event-data10	0x1D7A6	14806
event-data11	0x1D7B4	14820
event-data12	0x1D7C2	14834
event-data13	0x1D7D0	14848
event-data14	0x1D7DE	14862
event-data15	0x1D7EC	14876

`stack_ptr0` and `stack_ptr1` cycle the values shown starting from 14660

```
14666,14680,14694,14708,14722,14736,14750,14764,14778,14792,14806,14820,14834,
14848,14862,14876
```

Each event-data entry (*event_data0, event_data1, etc.*) is described as

```
bytes  datatype
---------------
  0-1     short   event-type (1=mousedown, 2=mouseup, 4=keyboard)
  2-3     short   key-id if the event is a keystroke
  4-5     short   mouse-button-id if the event is a mouse-press or mouse-release
                  (0 = left-button, 3 = right-button)
  6-9       int   the timestamp of the event
                  (the value of a global clock is written in when event is registered)
10-11     short   mouse cursor y-position [0,319]
12-13     short   mouse cursor x-position [0,189]
```

To simulate a mouse press or keystroke we can write these values at the position where `queue_ptr1` is pointing. After wrting the event data we finish the write-sequence by incrementing this pointer `queue_ptr1` (with the fixed values 14666,14680,14694,..,14876 and then back to 14666).

When the two pointers are different the game engine understands it as new mouse or keyboards events are available, if `queue_ptr0` is behind `queue_ptr1` it will process the event information pointed to by `queue_ptr0` (which points to the data just written to) and increment it until it equals `queue_ptr1`.