PlayStation Controllers

Taking advantage of specialized controllers and multi-player adapters

Using PlayStation Controllers

- Types of Controllers
- Reading Controller Data
- Detecting Various Controller Types
- Detecting & Using Multi-Player Adapters

Digital Controllers

- Sony Controller Pad
- Various 3rd-Party Pad Controllers
- Sony Mouse

Sony Controller Pad

- Shipped with every PlayStation
- 8 action buttons
  - L1, L2, R1, & R2
  - □, ○, △, & ×
- “Select” and “Start” buttons
- 4 button (8 direction) D-pad

Various 3rd Party Pad/Digital Joystick Controllers

- Examples
  - Alps Interactive Stingray
  - ASCII Entertainment ASCII Pad PS
  - ASCII Entertainment ASCII Stick PS
  - Interact Accessories PS Arcade
  - Interact Accessories PS ProPad
  - Optec Commander Pro

Various 3rd Party Pad/Digital Joystick Controllers

- No special support required in games
- Often have special buttons
  - “Turbo” variable-speed auto-fire
  - “Programmable” sequence buttons
    - Allows programming of special combo moves

Sony Mouse

- Two Button Mouse
- Returns movement as a vector from previous position
  - +8x, +4y
  - -3x, +2y
  - -2x, -2y
- Typically used as second controller
  - No “Start” or “Select” buttons, etc.

Analog Controllers

- Analog Joysticks
- Steering Controllers
- Namco neGcon
- Sony Analog Controller (SCPH-1110)
  - Double analog joysticks with D-pad hat
- Light Gun

Analog Controllers

- Namco neGcon
  - Digital joypad
    - Same as standard Sony Controller Pad
  - Analog “twist” channel
    - Works well as left / right for driving games
  - Three analog pressure-sensitive buttons
    - Typically used for throttle, brake, etc.
  - “Start” button

Analog Controllers

- Mad Catz Steering Controller
  - Left / Right analog channel
    - Same as neGcon “twist” channel
  - Separate analog “accelerator” and “brake” channels for plug-in pedal controllers
  - Standard digital “pad” controller
  - Six digital buttons
  - “Start” button

Analog Controllers

- Sony Analog Controller
  - Two analog joysticks
    - Left stick has L1, L2, R1 & R2 buttons
    - Right stick has  buttons
  - 8 buttons on base duplicate joystick buttons
  - Digital D-pad “hat” on right stick
    - Same as standard controller D-pad
  - “Start” & “Select” buttons

Analog Controllers

- Light Gun
  - Returns position of screen aimed at.
  - Trigger same as □ button
  - 2nd button same as ✕ button
  - “Start” button

Reading Controller Data

- Initialize Controller Reading
  - ***InitPad***(*buf1addr*, 8, *buf2addr*, 8)
    - Provide a buffer for each port, specify length
  - ***StartPad***
    - Enable controller reading
- Read Controller Data
  - Access contents of *buf1addr* & *buf2addr*

Detecting Various Controller Types

- All controllers return a two-byte header followed by 2-6 bytes of data
  - Controller ID specified in byte 1 of header
    - High nybble is type code
    - Low nybble is # of data bytes / 2
  - Controller input values in bytes 2-7

Byte	0	1	2	3	4	5	6	7
Contents	Status	ID	Button Data	Button Data	Analog channel A	Analog channel B	Analog channel C	Analog channel D

Controller Type Codes

Controller	Type Code
Sony Mouse	0x12
Analog Joystick	0x22
Driving Controller	0x23
Namco neGcon	0x23
Light Gun	0x31
Sony Controller Pad Third-party pad controllers	0x41
Sony Analog Controller (two stick Sony model SCPH-1110)	0x53
Sony Multi Tap multi-player adapter	0x80

Digital Button Data

Controller Type	Controller Data Buffer Byte 2	Controller Data Buffer Byte 3
Mouse	not used	Bit 2 = right Bit 3 = left
Others **(Some buttons not available on certain controller types)**	Bit 7 = D-pad Left Bit 6 = D-pad Down Bit 5 = D-pad Right Bit 4 = D-pad Up Bit 3 = Start Button Bit 2 = not used Bit 1 = not used Bit 0 = Select Button	Bit 7 = □ Button Bit 6 = X Button Bit 5 = △ Button Bit 4 = O Button Bit 3 = L1 Button Bit 2 = R1 Button Bit 1 = L2 Button Bit 0 = R2 Button

Analog Channel A

- X-axis position
  - Left hand joystick of Sony Analog Controller
    - Value range -128 to 127
- X-axis position
  - Analog Joysticks
    - Value range -128 to 127
- X-axis delta from previous position
  - Sony Mouse
    - Value range -128 to 127

Analog Channel A (continued)

- Steering Wheel Position
  - Driving Controllers
    - Value range 0-255
    - Center position = 128 (+/- 8) when released
- Twist Value
  - Namco neGcon
    - Value range 0-255
    - Center position = 128 (+/- 8) when released

Analog Channel B

- Y-axis position
  - Left hand joystick of Sony Analog Controller
    - value range -128 to 127
- Y-axis position
  - Analog Joysticks
    - Value range -128 to 127

Analog Channel B (continued)

- Throttle value for driving controllers
  - value range 0 to 255
- Y-axis delta from previous position
  - Sony Mouse
    - value range -128 to 127

Analog Channel C

- X-axis movement
  - Right hand joystick of Sony Analog Controller
    - Value Range -128 to 127
- Brake for driving controllers
    - Value range 0-255
- Pressure-sensitive button data for other controllers

Analog Channel D

- Y-axis movement
  - Right hand joystick of Sony Analog Controller
    - Value Range -128 to 127
- Pressure-sensitive button data for other controllers
    - Value Range (-128 to 127) or (0-255)

Analog Channel Hardware Notes

- Minimum & Maximum values
  - Values for throttle, brake, pressure-sensitive buttons should be no more than 16 when released, no less than 240 when fully depressed.
- Calibration Required
  - When using analog joysticks and steering wheels, provide a calibration screen to determine center position, minimum and maximum values.

Multi-Controller Adapters

- Sony Multi Tap multi-controller adapter
  - Connects up to 4 controllers
  - Connects up to memory cards
- Detecting The Multi Tap
- Using The Multi Tap

Detecting the Multi Tap Adapter

- Must use Multi Tap library
  - Otherwise PlayStation only sees controller in Multi Tap port A.
- Multi-Tap returns controller ID 0x80
  - Multi Tap status byte and controller ID are followed by 4 packets of 8 bytes each containing controller data for each port of the Multi Tap.

Using the Multi Tap

- Returns 34 bytes per port, instead of 8 bytes
  - Specify larger buffer with ***InitTap***
  - 8 bytes of data per port, as discussed earlier
- Unused ports return controller ID 0x00

Bytes	0	1	2-10	11-18	19-26	27-35
Contents	Status	0x80 Multi Tap ID code	Data for controller port A on Multi Tap	Data for controller port B on Multi Tap	Data for controller port C on Multi Tap	Data for controller port D on Multi Tap

Initializing the Light Gun

- Initialize event handling needed for gun interrupt

```
ev0 = OpenEvent(SwCARD, EvSpIOE, EvMdNOINTR, NULL);
ev1 = OpenEvent(SwCARD, EvSpERROR, EvMdNOINTR, NULL);
ev2 = OpenEvent(SwCARD, EvSpTIMOUT, EvMdNOINTR, NULL);
ev3 = OpenEvent(SwCARD, EvSpNEW, EvMdNOINTR, NULL);
ev10 = OpenEvent(HwCARD, EvSpIOE, EvMdNOINTR, NULL);
ev11 = OpenEvent(HwCARD, EvSpERROR, EvMdNOINTR, NULL);
ev12 = OpenEvent(HwCARD, EvSpTIMOUT, EvMdNOINTR, NULL);
ev13 = OpenEvent(HwCARD, EvSpNEW, EvMdNOINTR, NULL);
```

Initializing the Light Gun

- Initialize controller reading & data buffer with ***InitPad***
  - Specify an 86 byte buffer instead of an 8-byte buffer to accommodate larger amount of data returned by gun.

```
#define PADBUFFLEN (86)

InitPAD(      &p.buff[0][0], PADBUFFLEN,
              &p.buff[1][0], PADBUFFLEN );
```

Initializing the Light Gun

- Initialize card services (used for the gun interrupt handling)

```
InitCARD(0);        /* reset PAD */
StartCARD();
```

Initializing the Light Gun

- Use ***InitGun*** to specify the location and length of the buffer that will receive gun position data
  - Length is specified as number of X/Y pairs
    - Length of 20 = 80 bytes

```
InitGun(&p.buff[0][4], &p.buff[1][4], 20);
```

Initializing the Light Gun

- Enable controller reading using ***StartPad*** and ***StartGun***

```
StartPAD();
StartGun();
```

Initializing the Light Gun

- Enable gun events

```
EnableEvent(ev0);
EnableEvent(ev1);
EnableEvent(ev2);
EnableEvent(ev3);
EnableEvent(ev10);
EnableEvent(ev11);
EnableEvent(ev12);
EnableEvent(ev13);
```

Initializing the Light Gun

- Select which port to enable for gun reading using the ***SelectGun*** function.
  - ***SelectGun***(int *port*, int *enable_flag*)
    - ***SelectGun***(0,1) selects port 0
    - ***SelectGun***(1,1) selects port 1
    - Only one port can be enabled each video field
      - Reading each gun at 30fps works OK.

```
SelectGun(0,1);
```

Reading the Light Gun

- Controller data buffer specified with ***InitPad*** contains gun status, ID code, and button data

Buffer Position	0	1	2	3
Value	Gun status	0x31 Light Gun ID code	Bit 3 = Start button	Bit 7 = Trigger (□ Button) Bit 6 = Button 2 (× Button)

Reading the Light Gun

- Buffer specified with ***InitGun*** contains gun position coordinate pairs

Buffer Position	0	1	2 & 3	4 & 5	6 & 7	8 & 9	…
Value	0 (unused)	Count of X/Y pairs in rest of buffer	X-axis position value #0	Y-axis position value #0	X-axis position value #1 (if specified by count)	Y-axis position value #1 (if specified by count)	etc. (if specified by count)

Reading the Light Gun

- Position values are specified as hardware ticks & require some post-processing.
  - Use following methods as starting point, but you may need to fine tune your application

Reading the Light Gun • Post-processing The Position Values

- X-Axis value is the horizontal pixel clock value where gun detected the electron beam.
  - x = x - 140
    if(x > 0)
    x = (x * 1000) / screen width divisor
    else
    x = 0
  - Divisor = 5000 for 320-pixel wide display
  - Divisor = 3125 for 512-pixel wide display
  - Divisor = 2500 for 640-pixel wide display

Reading the Light Gun

- Y-Axis value is the # scanlines since vertical blank, including those above the display area
  - Y = Y - 16 (actually, this is the Y-offset specified to GPU)
  - Multiply by 1 for 240-line screen
  - Multiply by 2 for 480-line screen

Gun Considerations

- High-Priority DMA operations can reduce light gun accuracy
  - Horizontal position of electron beam changes too much before interrupt occurs and gun position is saved, causes inaccuracy.
  - Happens when doing big chunks of DMA at high priority, which ties up bus for too long
    - Streaming CD-ROM reading
    - MDEC decoding

Gun Considerations

- Light Gun library and Multi Tap library are currently incompatible
  - Fixed in library v3.6 (due Oct.)
- On the other hand…
  - Can only read 1 gun per frame anyway, so having more than 2 guns would really reduce the response time for each gun.

The End