CS 134

Flexible Lecture:
Alternate Inputs

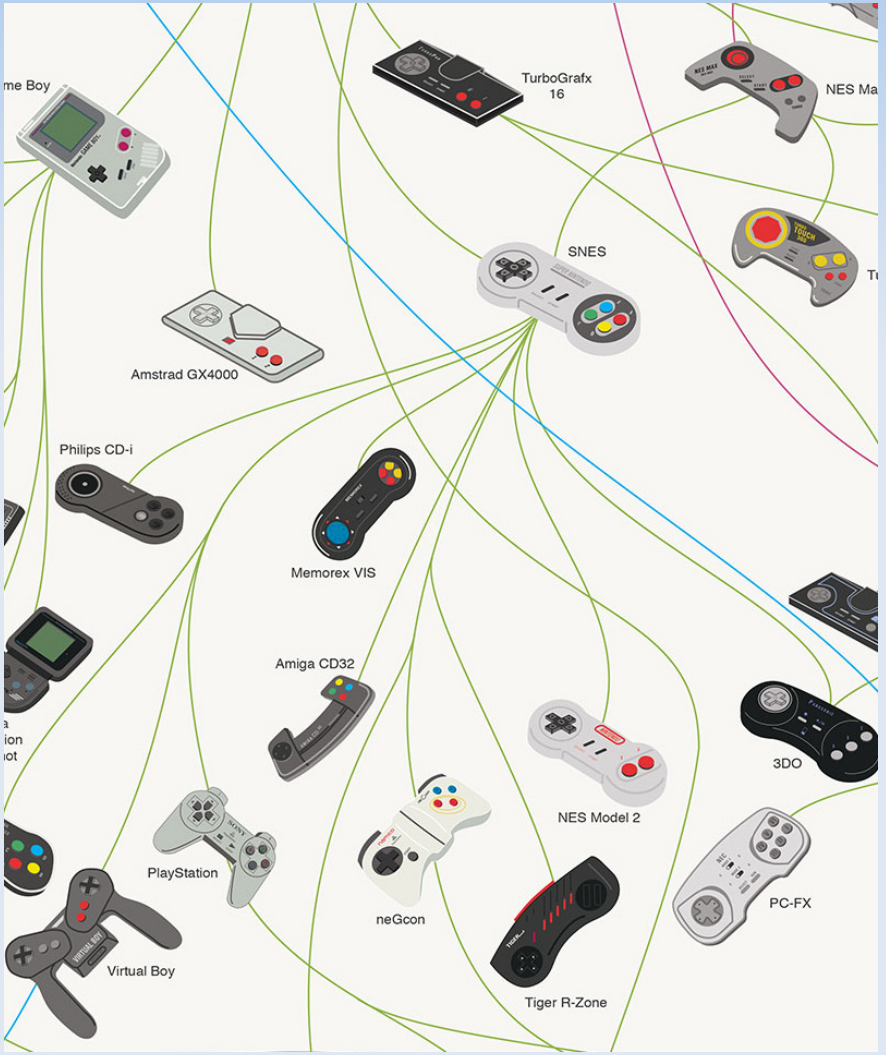
Updates

Send me requests for the Flexi-Lectures

Everyone should be on a team now
Final project due in 28 days!

Alternate Inputs

- This class:
  - SDL_Joystick
  - Touchscreen



XInput

- Only one thing...

- Get more information from
  - https://msdn.microsoft.com/en-us/library/windows/desktop/hh405051%28v=vs.85%29.aspx

SDL_Joystick

SDL_Joystick

- Designed to support arbitrary gamepads, joysticks, and HOTAS
- Extremely generic and very inconsistent
  - Axis 1 is USUALLY X
  - Axis 2 is USUALLY Y
  - Everything else fair game

SDL_Joystick

- Setup / Cleanup
  - Need to init with SDL_INIT_JOYSTICK
  - SDL_JoystickOpen() / SDL_JoystickClose()
- API
  - SDL_JoystickGetAxis()
  - SDL_JoystickGetHat()
  - SDL_JoystickGetButton()
  - And SDL_JoystickNumXXX() variants
- Gestures
  - Deadzone for axes

SDL_Joystick

- At initialization time, you need to pass SDL_INIT_JOYSTICK to SDL_Init().
- Before using a joystick, you must open it.
  - int SDL_NumJoysticks()
  - SDL_Joystick* SDL_JoystickOpen(int index)
- When done, you can close it, but you don't need to
  - void SDL_JoystickClose(SDL_Joystick*)

SDL_Joystick

- Support arbitrary joysticks, so you need to figure out what is available to query
  - Axis – from -32768 to 32767, 0 by default
  - Hat – centered, up, down, left, right, etc.
  - Button – on or off
  - Ball – a mini mouse on the joystick
- For a given joystick you can ask how many
  - int SDL_JoystickNumXXX(SDL_Joystick*)

SDL_Joystick

- To read from a specific axis/button/hat/ball you need to ask about a specific one

- SDL_JoystickGetAxis(SDL_Joystick*, int)
- SDL_JoystickGetHat(SDL_Joystick*, int)
- SDL_JoystickGetButton(SDL_Joystick*, int)
- SDL_JoystickGetBall(SDL_Joystick*, int,
  int* dx, int* dy)

SDL_Joystick

- Because of all the genericness, you will want to provide a simpler interface, more like XInput
- struct MyGamepad {

```
    int leftStickX, leftStickY;
    int rightStickX, rightStickY;
    int a, b, x, y;
}
```

- Set up this data every frame!

SDL_Joystick

```
// globals
SDL_Joystick joy1;
MyGamepad gamepad;

// game loop
    while( !shouldExit ) {
        /* next to all the other input handling */
        gamepad.leftStickX = SDL_JoystickGetAxis( joy1, 0 );
        gamepad.leftStickY = SDL_JoystickGetAxis( joy1, 1 );
        gamepad.rightStickX = SDL_JoystickGetAxis( joy1, 2 );
        gamepad.rightStickY = SDL_JoystickGetAxis( joy1, 3 );
        gamepad.a = SDL_JoystickGetButton( joy1, 0 );
        gamepad.b = SDL_JoystickGetButton( joy1, 1 );
        gamepad.x = SDL_JoystickGetButton( joy1, 2 );
        gamepad.y = SDL_JoystickGetButton( joy1, 3 );

        /* Rest of the game loop goes here */

        SDL_GL_SwapWindow( window );
    }
```

SDL_Joystick

- Beware though, there's no real consistency
  - Sometimes a D-Pad is four buttons
  - Sometimes a D-Pad is a hat
  - Sometimes a D-Pad is two axes
- If your game requires more than one stick and four buttons, you really will want to put a customization layer in.
  - For each input, store which button / axis / hat to use.

SDL_Joystick

Questions?

Touchscreen

Touchscreen

This is just ONE of many things needed to support Android or iOS.

Touchscreen

- No setup or cleanup code
- API
  - SDL_FINGERDOWN
  - SDL_FINGERMOTION
  - SDL_FINGERUP
- Gestures
  - Tap, Long press, Pinch

Touchscreen

- Touchscreen API is entirely through the message pump
- Like with the Joystick interface, you will want to store the results in globals so your update logic can look at the data

Touchscreen

- Event types:
  - SDL_FINGERDOWN – Just went down
  - SDL_FINGERUP – Just went up
  - SDL_FINGERMOTION – Down and moving
- Event fields:
  - .tfinger.fingerId – Finger being talked about
  - .tfinger.x – 0-1 position of the finger
  - .tfinger.y – 0-1 position of the finger

Touchscreen

- For your game, decide how many fingers you will support max
- In input logic, update the finger state based on the events
- fingerID does not need to start at 0!
- Per finger state:
  - bool isDown
  - float x
  - float y

Touchscreen

```
// globals
bool fingerIDSet;
SDL_FingerID fingerID;
bool fingerDown;
float fingerX;
float fingerY;

// message pump
        while( SDL_PollEvent( &event )) {
            switch( event.type ) {
                case SDL_QUIT:
                    shouldExit = 1;
                    break;
                case SDL_FINGERDOWN:
                    if(!fingerIDSet) {
                        fingerID = event.tfinger.fingerId;
                    }
                    if(event.tfinger.fingerID != fingerID) {
                        break;
                    }
                    fingerDown = true;
                    fingerX = event.tfinger.fingerX;
                    fingerY = event.tfinger.fingerY;
            }
        }
```

Touchscreen

```
        case SDL_FINGERUP:
            if(!fingerIDSet) {
                fingerID = event.tfinger.fingerId;
            }
            if(event.tfinger.fingerID != fingerID) {
                break;
            }
            fingerDown = false;
            fingerX = event.tfinger.fingerX;
            fingerY = event.tfinger.fingerY;
        case SDL_FINGERMOTION:
            if(!fingerIDSet) {
                fingerID = event.tfinger.fingerId;
            }
            if(event.tfinger.fingerID != fingerID) {
                break;
            }
            fingerDown = true;
            fingerX = event.tfinger.fingerX;
            fingerY = event.tfinger.fingerY;
    }
}
```

Touchscreen

- Touchscreen gestures should be handled just like the mouse gestures

- Tap is like click
- Longpress is like drag

Touchscreen

- Touchscreen gestures should be handled just like the mouse gestures

- Tap is like click
- Longpress is like drag

Touchscreen

Questions?

Touchscreen

- Touchscreen gestures should be handled just like the mouse gestures

- Tap is like click
- Longpress is like drag