Ryan C. Gordon

icculus.org

Game Development with SDL 2.0

A few notes…

- Feel free to interrupt!
- Slides are at https://icculus.org/SteamDevDays/
- Today is a high-level overview.
- Feel free to tweet at @icculus

Who am I?

- Hacker, game developer, porter
- Port games, build tools
- Freelance
- 15 years experience

loki
ACTIVISION
Google
CRYTEK
humanhead studios
DOUBLE FINE
TEAM MEAT
RED MARBLE GAMES
Gaslamp Games
mode7
EPIC GAMES
LARGE ANIMAL GAMES
imvu
VALVE
klei
klei entertainment
MINMAX GAMES
CROTEAM
TRIPWIRE INTERACTIVE
TRENDY ENTERTAINMENT
dice
PopCap
Spiderweb SOFTWARE
GUNNARGAMES
kranx productions
Reflexive entertainment
BIT BLOT
EA GAMES
aspyr
LAZY 8 STUDIOS
oberon media

Class
of
2030

What is SDL?

- Simple Directmedia Layer
- Open source answer to DirectX.
- Cross-platform, powerful, fast, easy.
- 15 years of development.
- Many games, millions of gamers.
- https://www.libsdl.org/

History

- Started by Sam Lantinga for Executor.
- Used by Loki Software for Linux titles.
- Now a de facto standard.
- SDL 2.0 is the new hotness.

Features

- Modern OSes and devices
- Portable game framework
- Multiple API targets
- Makes hard things easy
- Written in C
- zlib licensed

Platforms

- Linux
- Mac OS X
- Windows
- Unix
- Android
- iOS
- Haiku
- Raspberry Pi
- Other interesting places

Subsystems

- Audio
- Events
- Rendering
- Joystick
- Game Controllers
- Haptic
- Shared Libraries
- CPU Info
- Stdlib
- Timers
- Threads
- RWops

Pick Your Target

- Runtime choice with dlopen()
- X11, Wayland, Mir…
- ALSA, PulseAudio, OSS, esd, arts, nas…
- winmm, DirectSound, XAudio2…

Dirt-simple Direct3D example

```
WNDCLASSEX winClass;
MSG        uMsg;

 memset(&uMsg,0,sizeof(uMsg));

winClass.lpszClassName = "MY_WINDOWS_CLASS";
winClass.cbSize        = sizeof(WNDCLASSEX);
winClass.style         = CS_HREDRAW | CS_VREDRAW;
winClass.lpfnWndProc   = WindowProc;
winClass.hInstance     = hInstance;
winClass.hIcon          = LoadIcon(hInstance, (LPCTSTR)IDI_DIRECTX_ICON);
 winClass.hIconSm     = LoadIcon(hInstance, (LPCTSTR)IDI_DIRECTX_ICON);
winClass.hCursor       = LoadCursor(NULL, IDC_ARROW);
winClass.hbrBackground = (HBRUSH)GetStockObject(BLACK_BRUSH);
winClass.lpszMenuName  = NULL;
winClass.cbClsExtra    = 0;
winClass.cbWndExtra    = 0;

if( RegisterClassEx(&winClass) == 0 )
   return E_FAIL;

g_hWnd = CreateWindowEx( NULL, "MY_WINDOWS_CLASS",
                          "Direct3D (DX9) - Full Screen",
                 WS_POPUP | WS_SYSMENU | WS_VISIBLE,
                   0, 0, 640, 480, NULL, NULL, hInstance, NULL );

if( g_hWnd == NULL )
   return E_FAIL;

 ShowWindow( g_hWnd, nCmdShow );
 UpdateWindow( g_hWnd );
```

```
g_pD3D = Direct3DCreate9( D3D_SDK_VERSION );

if( g_pD3D == NULL )
{
    // TO DO: Respond to failure of Direct3DCreate8
    return;
}

//
// For the default adapter, examine all of its display modes to see if any
// of them can give us the hardware support we desire.
//

int nMode = 0;
D3DDISPLAYMODE d3ddm;
bool bDesiredAdapterModeFound = false;

int nMaxAdapterModes = g_pD3D->GetAdapterModeCount( D3DADAPTER_DEFAULT,
                                                     D3DFMT_X8R8G8B8 );

for( nMode = 0; nMode < nMaxAdapterModes; ++nMode )
{
    if( FAILED( g_pD3D->EnumAdapterModes( D3DADAPTER_DEFAULT,
                                          D3DFMT_X8R8G8B8, nMode, &d3ddm ) ) )
    {
        // TO DO: Respond to failure of EnumAdapterModes
        return;
    }

    // Does this adapter mode support a mode of 640 x 480?
   if( d3ddm.Width != 640 || d3ddm.Height != 480 )
       continue;

    // Does this adapter mode support a 32-bit RGB pixel format?
    if( d3ddm.Format != D3DFMT_X8R8G8B8 )
       continue;

    // Does this adapter mode support a refresh rate of 75 MHz?
    if( d3ddm.RefreshRate != 75 )
        continue;

    // We found a match!
    bDesiredAdapterModeFound = true;
    break;
}

if( bDesiredAdapterModeFound == false )
{
    // TO DO: Handle lack of support for desired adapter mode...
    return;
}
```

```
// Can we get a 32-bit back buffer?
if( FAILED( g_pD3D->CheckDeviceType( D3DADAPTER_DEFAULT,
                                     D3DDEVTYPE_HAL,
                                     D3DFMT_X8R8G8B8,
                                     D3DFMT_X8R8G8B8,
                                     FALSE ) ) )
{
    // TO DO: Handle lack of support for a 32-bit back buffer...
    return;
}

// Can we get a z-buffer that's at least 16 bits?
if( FAILED( g_pD3D->CheckDeviceFormat( D3DADAPTER_DEFAULT,
                                       D3DDEVTYPE_HAL,
                                               D3DFMT_X8R8G8B8,
                                       D3DUSAGE_DEPTHSTENCIL,
                                       D3DRTYPE_SURFACE,
                                       D3DFMT_D16 ) ) )
{
    // TO DO: Handle lack of support for a 16-bit z-buffer...
    return;
}

//
// Do we support hardware vertex processing? if so, use it.
// If not, downgrade to software.
//

D3DCAPS9 d3dCaps;

if( FAILED( g_pD3D->GetDeviceCaps( D3DADAPTER_DEFAULT,
                                   D3DDEVTYPE_HAL, &d3dCaps ) ) )
{
    // TO DO: Respond to failure of GetDeviceCaps
    return;
}

DWORD flags = 0;

if( d3dCaps.VertexProcessingCaps != 0 )
    flags = D3DCREATE_HARDWARE_VERTEXPROCESSING;
else
    flags = D3DCREATE_SOFTWARE_VERTEXPROCESSING;
```

```
//
// Everything checks out - create a simple, full-screen device.
//

D3DPRESENT_PARAMETERS d3dpp;
memset(&d3dpp, 0, sizeof(d3dpp));

d3dpp.Windowed               = FALSE;
d3dpp.EnableAutoDepthStencil = TRUE;
d3dpp.AutoDepthStencilFormat = D3DFMT_D16;
d3dpp.SwapEffect             = D3DSWAPEFFECT_DISCARD;
d3dpp.BackBufferWidth        = 640;
 d3dpp.BackBufferHeight       = 480;
 d3dpp.BackBufferFormat       = D3DFMT_X8R8G8B8;
 d3dpp.PresentationInterval   = D3DPRESENT_INTERVAL_IMMEDIATE;

if( FAILED( g_pD3D->CreateDevice( D3DADAPTER_DEFAULT, D3DDEVTYPE_HAL, g_hWnd,
                                            flags, &d3dpp, &g_pd3dDevice ) ) )
{
     // TO DO: Respond to failure of CreateDevice
     return;
}
```

```
// TO DO: Respond to failure of Direct3DCreate8
```

Really hard SDL version

```
SDL_Init(SDL_INIT_VIDEO);

SDL_CreateWindow(
    “Hello”, 0, 0, 640, 480,

SDL_WINDOW_FULLSCREEN |
    SDL_WINDOW_OPENGL
);
```

Video API

- Multiple windows, multiple displays
- Drawing: Software, OpenGL, GLES, Direct3D
- Makes OpenGL context management easy
- Exposes system GUI events
- Message boxes

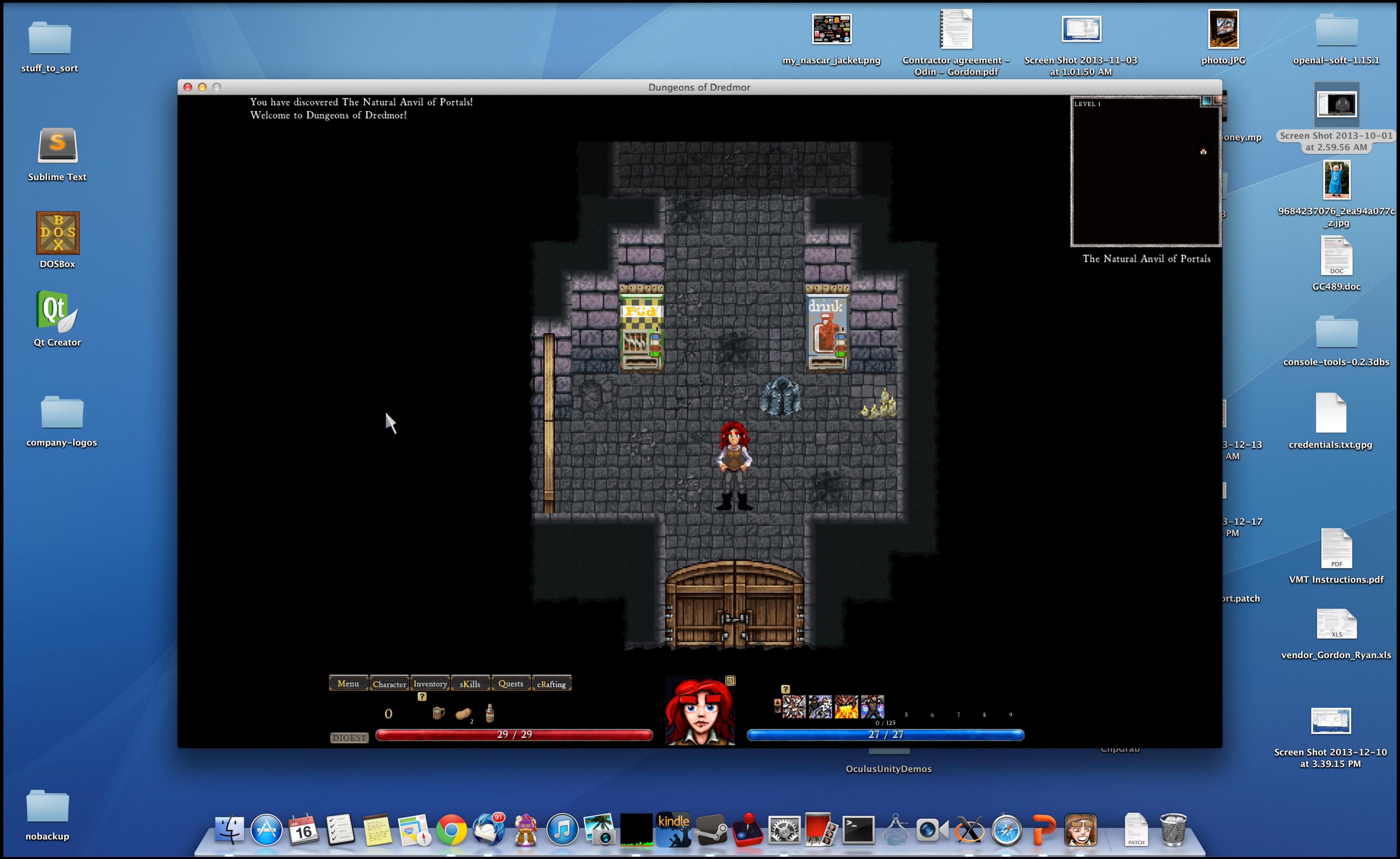
Video API Concepts

- Windows
- Surfaces
- Textures
- OpenGL, etc.

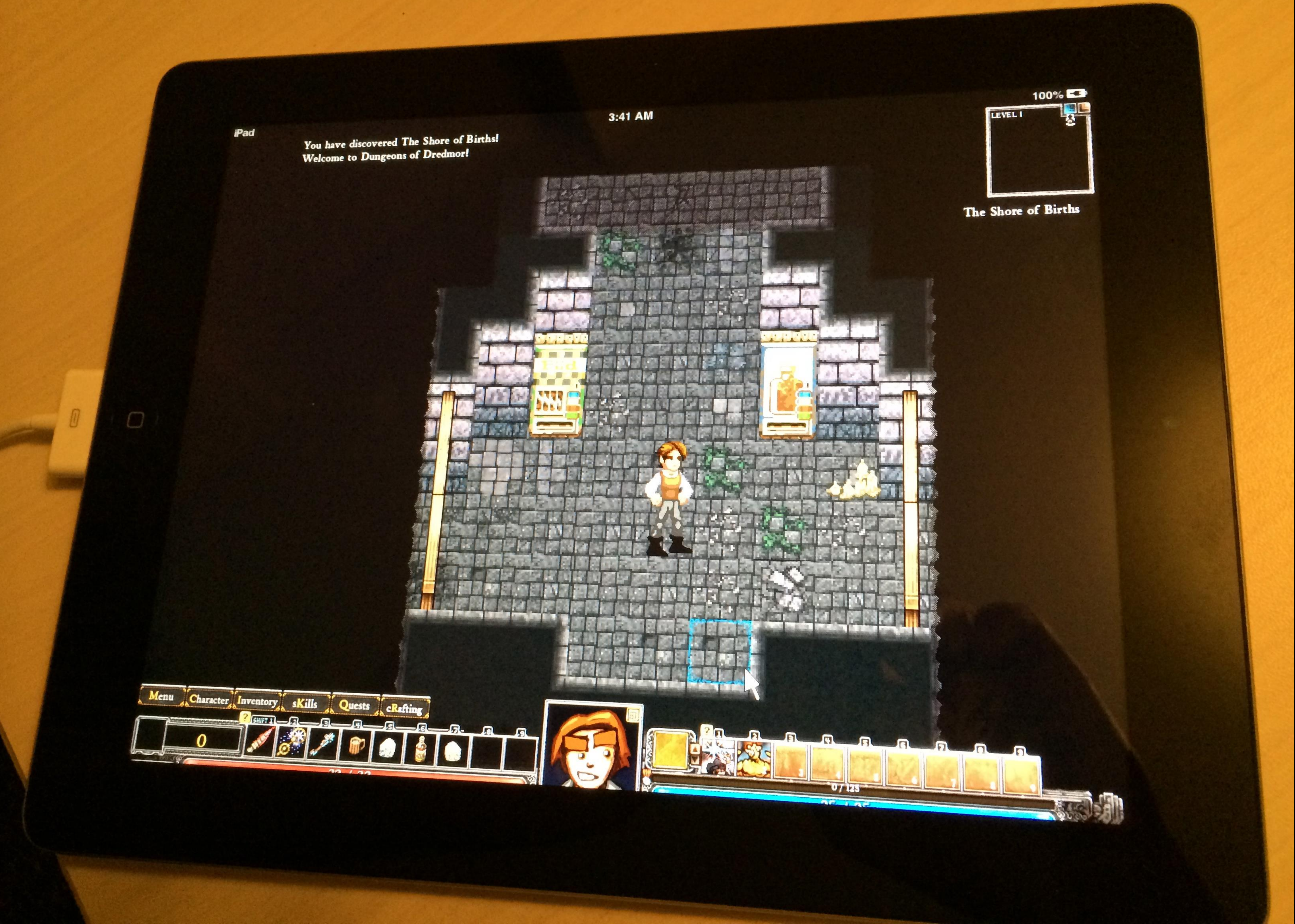
Render API

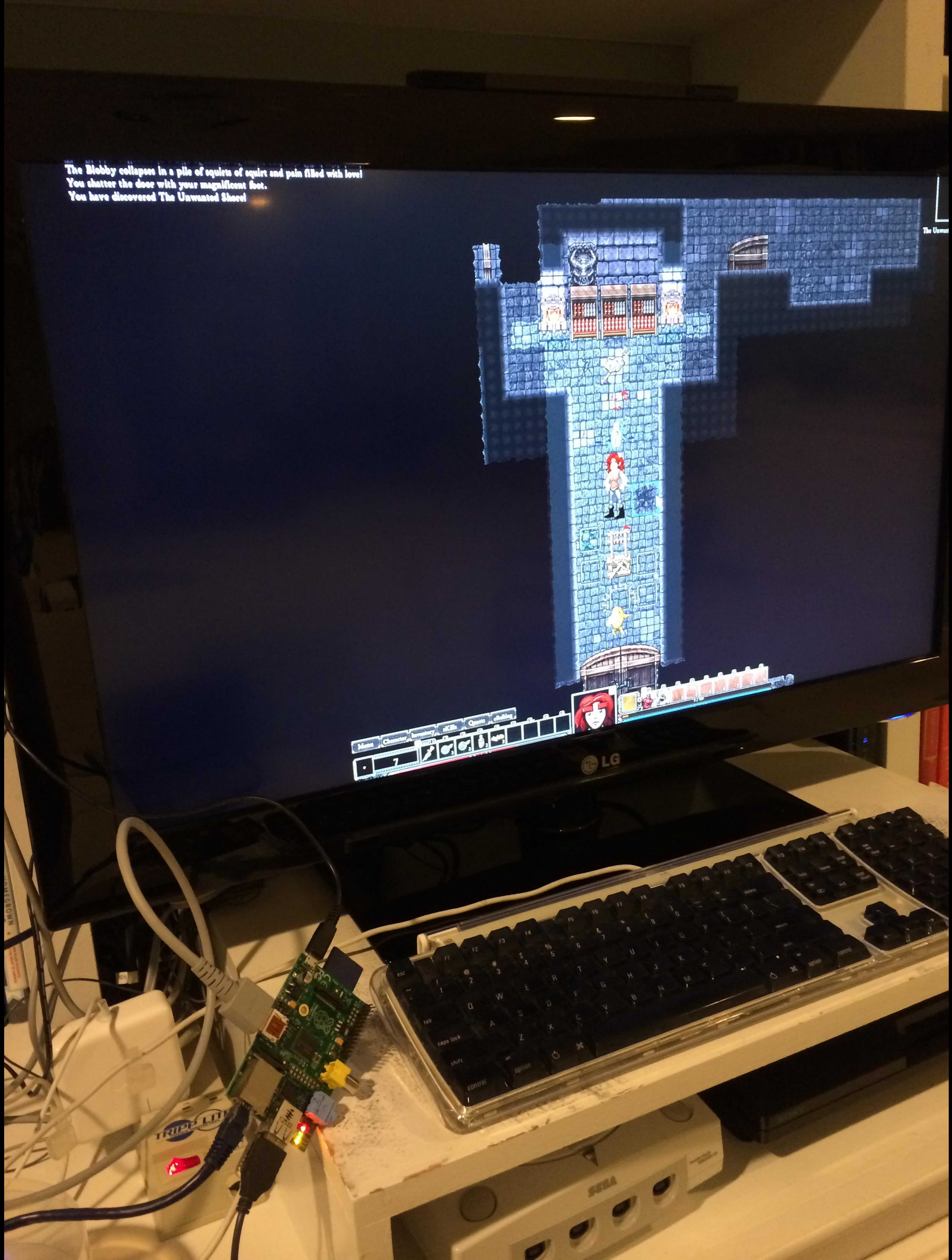
- Simple 2D API
- Backed by GPU
- Sprites, color ops, blending, primitives, scaling, rotation
- Write simple games fast
- Make legacy games amazing!
- Need more power? Use OpenGL.

Dungeons of Dredmor vs SDL2.



You have discovered The Natural Anvil of Portals!
Welcome to Dungeons of Dredmor!


iPad
3:41 AM
100%
You have discovered The Shore of Births!
Welcome to Dungeons of Dredmor!
LEVEL 1
The Shore of Births
Menu
Character
Inventory
sKills
Quests
cRafting
0
0 / 125


The Blobby collapses in a pile of squirts of squirt and pain filled with love!
You shatter the door with your magnificent foot.
You have discovered The Unwanted Shore!

Disclaimer

Gaslamp Games is not shipping these things, or planning to at the moment.
It was Just For Fun.

Using OpenGL

```
SDL_Init(SDL_INIT_VIDEO);
SDL_Window *win = SDL_CreateWindow(
    “Hello”, 0, 0, 640, 480, SDL_WINDOW_OPENGL);
SDL_GL_CreateContext(win);

// START MAKING OPENGL CALLS HERE.

SDL_GL_SwapWindow(win);
```

Events

- OS Events (mouse, keyboard, window)
- Relative mouse mode
- Touch API
- Gestures
- Joysticks and Game Controllers
- Timers

Event loop

```
SDL_Event event;
while (SDL_PollEvent(&event))
{
    switch (event.type) {
        case
SDL_MOUSEMOTION:
                // blah
        case SDL_KEYDOWN:
                // blah blah
        case SDL_QUIT:
                // bloop bleep
    }
}
```

Joystick API

- Multiple sticks
- Polling or events
- Query axes, buttons, hats, names
- Connect and disconnect notifications

Game Controller API

- Everything wants an XBox controller. (:( )
- Automatic configuration.
- Steam Big Picture support
- Crowd-sourced configurations
- Less flexible, but Just Works really well.

Haptic API

- “Haptic” == “Force feedback”
- Supports controllers *and* mice!
- Complex effects, simple rumble, left/right
- Fire and forget

Audio API

- VERY low-level. Maybe too low-level.
- Multiple devices, connect/disconnect
- Mono, Stereo, Quad, 5.1
- 8/16/32 bit, (un)signed, little/big, int/float
- On-the-fly conversion/resampling
- You feed us uncompressed PCM data in a callback.

Really, it’s low-level.

- Only a relentless stream of PCM.
- You mix, you spatialize, you manage.
- Try SDL_mixer or OpenAL.

Threading API

- SDL_CreateThread()
- Mutexes
- Semaphores
- Conditions
- Atomics

Other APIs

- Message Boxes
- Clipboard
- syswm
- CPU Info
- Stdlib
- Timers
- RWops
- Filesystems
- Power

The (Near) Future

- Multiple mice
- Audio capture, video capture
- 7.1 audio
- Wayland, Mir, libdrc
- WinRT and Windows Store apps
- sdl12_compat
- The Dynamic API
- Your requests here!

Getting involved

- Mailing lists! https://lists.libsdl.org/
- Forums! https://forums.libsdl.org/
- Wiki! https://wiki.libsdl.org/
- Bugs! https://bugzilla.libsdl.org/
- Buildbot! https://buildbot.libsdl.org/
- Everything else! https://www.libsdl.org/

That’s all folks.

- Questions? Answers!
- Hire me.
- https://icculus.org/SteamDevDays/
- Ryan C. Gordon: icculus@icculus.org
- https://twitter.com/icculus
- http://gplus.to/icculus