GDC | March 20-24, 2023
San Francisco, CA

SDL: Past, Present, and Future

Sam Lantinga and Ryan Gordon

#GDC23

What Is SDL?

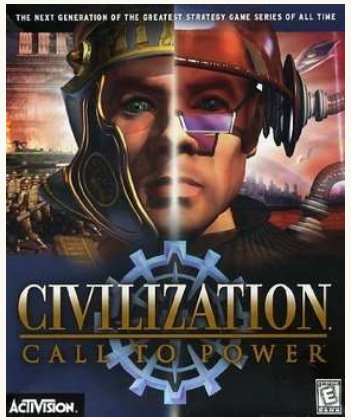
- Simple DirectMedia Layer
- Cross-platform interface for hardware abstraction
  - Windows, macOS, Linux, iOS, Android, etc.
- Covers audio, video, controllers, and much more
- Used by game engines and game/app developers
- No, my middle initial isn’t ‘D’

Who Are We?

- Sam Lantinga
  - Author of SDL, Software Engineer at Valve Corporation
- Ryan Gordon
  - Partner in crime, Software Engineer at icculus.org
- Unsung heroes
  - @1bsyl, @Kontrabant, @madebr, @sezero
- And many many more…

The Past: SDL 1.2

- Born out of Executor, a 68k Macintosh emulator
- Single window, framebuffer access, single audio callback, keyboard/mouse/joystick input, CD-ROM support
- Used by Loki Software in Linux ports



The Present: SDL 2.0

- Added support for multiple windows, surround sound, accelerated 2D video API, Vulkan graphics context, game controller API
- Used by Valve Corporation in Steam and the Source 2 engine across all platforms

25 Years of History

- Started with Loki Software in 1998
  - Civilization: Call To Power, Heroes of Might and Magic III, Sid Meier's Alpha Centauri, Tribes 2, Unreal Tournament, etc.
- Continues with Valve Corporation in 2023
  - DOTA 2, Half-Life: Alyx, Portal 2, Steam, Steam Link, Steam VR, Team Fortress 2, etc.
- Used by hundreds of games, including many published on Steam
- Used by several major game engines, including Unity and Source 2
- Used by games, emulators, embedded environments, and more!
- Has bindings for over a dozen programming languages

25 Years of Portability

- Windows, Linux, *BSD, macOS, iOS, tvOS, Android
- Windows RT, Windows UWP, Windows and Xbox GDK
- Playstation, Nintendo Switch
- Emscripten: reimplemented SDL 1.2 API using Javascript, now directly supported in SDL 2.0
- And many more over the course of time… Haiku, N3DS, OS/2, PS Vita, QNX, etc.

Open Source Is Great

- Moving to GitHub has greatly increased developer engagement and feedback, and has directly contributed to the move to SDL 3
- There are over 400 contributors and 1000 forks of the core library
- Moving to GitHub has made it much easier to release quickly, and now we have a monthly cadence of stable release updates

Commercial Products Are Great

- Great feedback from professional developers shipping large products
- Battle tested, shipping in Steam to millions of customers every day
- Used by Steam Link exclusively for audio, video, and input across mobile and desktop platforms
- Shipping SDL pre-release code in Steam betas results in more stable releases

The Future: SDL 3.0

- Opportunity to apply what we've learned over the last 10 years, changing the API and ABI
- Simplify and streamline the build process
- Simplify and streamline the API
- Add new useful functionality

The Future: SDL 3.0

- Simplifying and streamlining the build process
  - Standardizing on Cmake
  - Visual Studio and Xcode projects still available

The Future: SDL 3.0

- Simplifying and streamlining the API
  - Symbol naming conventions are more consistent
  - Simplifying functions where it makes sense
  - Removing functions where it makes sense
  - main() handling moved to a standalone header library

The Future: SDL 3.0

- Adding new useful functionality
  - New 3D GPU API in development
  - Full support for high DPI displays
  - Added nanosecond time precision
  - Added sub-frame event timing
  - And much, much more!

The Compatibility Story

- Compatibility is important so older commercial games continue to run as platforms evolve
- sdl12-compat allows running SDL 1.2 games on the SDL 2.0 runtime
- sdl2-compat allows running SDL 2.0 games on the SDL 3.0 runtime
- SDL 1.2 => sdl12-compat => sdl2-compat => SDL 3.0 works!
- Civilization: Call To Power from 1999 runs seamlessly on a modern Linux system running Wayland

Easing the Transition

- Careful decisions about how to change the codebase
  - Code style reformatting was applied to both SDL2 and SDL3 so bug fixes could be more easily merged between major versions
  - We decided not to switch to stdint types for internal code

Easing the Transition

- Careful decisions about how to change the API
  - Does the change make life better for developers?
  - Are we changing the API in a way that will still make sense 10 years from now?

Easing the Transition

- Transition guide
  - As part of the change acceptance process, each change must be documented and a developer transition plan provided in docs/README-migration.md
  - Coccinelle is a tool that can be used on Linux or on Windows via WSL

GDC | March 20-24, 2023
San Francisco, CA

Topic: Relative Mouse Motion

#GDC23

Topic: Relative Mouse Motion

- When is it useful?
  - Camera control in FPS style games
  - Dragging the map in RTS or RPG style games
  - Precise mouse positioning in emulators
  - … etc.

Topic: Relative Mouse Motion

- Classic approach:
  - Mouse warping - originally used in the id DOOM engine
  - Generates additional mouse events
  - Deltas are affected by desktop mouse acceleration
  - Doesn’t work over Windows Remote Desktop

Topic: Relative Mouse Motion

- SDL_SetRelativeMouseMode()
  - Uses raw input instead of mouse warping
  - Automatically hides the mouse cursor
  - Automatically constrains the mouse to the window
  - Provides low level, low latency mouse deltas
  - Delta scaling disabled by default for predictable movement
  - Works with Windows Remote Desktop (sorta)

Topic: Relative Mouse Motion

- Considerations
  - Warping the mouse in relative mode does not generate a mouse event, but does change mouse position
  - You can save and restore mouse position when enabling and disabling relative mouse mode
  - You can change relative motion sensitivity by setting SDL_HINT_MOUSE_RELATIVE_SPEED_SCALE
  - You can enable desktop mouse acceleration curves in relative mode by setting SDL_HINT_MOUSE_RELATIVE_SYSTEM_SCALE

Conclusion: Relative Mouse Motion

- Don’t use mouse warping!
- SDL_SetRelativeMouseMode() is a great way to easily add relative mouse motion to your application

GDC
March 20-24, 2023
San Francisco, CA

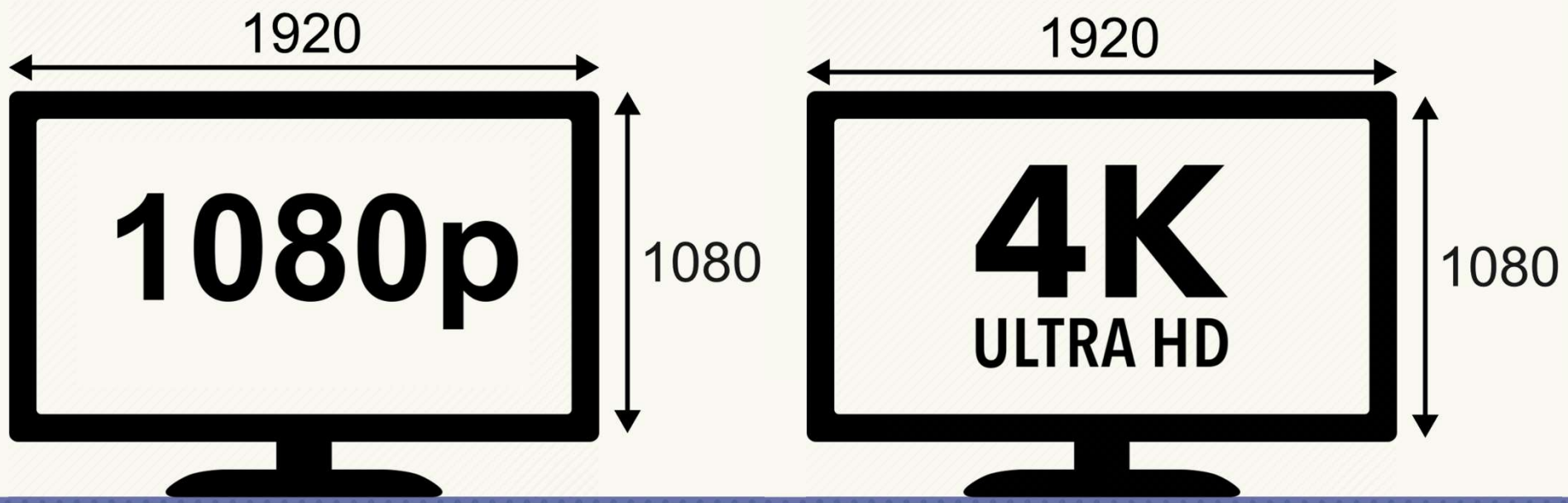
Topic: High DPI Support

Topic: High DPI Support

- A tale of screen coordinates, scale, and pixels…
  - Screen coordinates, also known as points or device independent pixels, are the size in pixels divided by the display scaling factor
  - Screen coordinates are oriented around content physical size
  - More pixels means more detail, but not more content

Topic: High DPI Support

- A 4K display with 200% scaling has 1080p screen coordinate size



Topic: High DPI Support

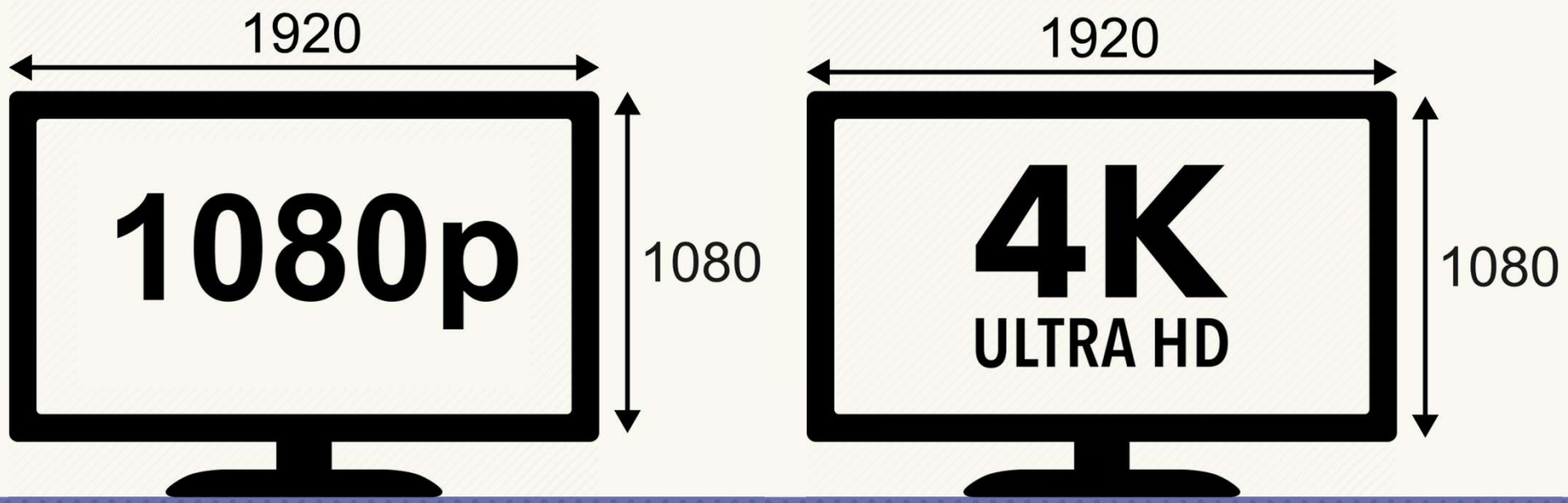
- Apple platforms provide screen coordinates, optionally allows high DPI back buffers
- Windows provides screen coordinates for non-DPI aware applications, pixels for DPI aware applications

Topic: High DPI Support

- SDL 3.0 applications always get screen coordinates and high DPI back buffers
- Display bounds, window coordinates, and mouse coordinates are all specified in screen coordinates
- Mouse coordinates are floating point with full precision, centered on pixels
- Display modes include size in screen coordinates, size in pixels, and a display scale

Topic: High DPI Support

- A fullscreen desktop window is 1920x1080 in both cases, and a 4K back buffer on the 4K monitor



Topic: High DPI Support

- SDL_GetDesktopDisplayMode() returns the current desktop mode, including scale
- SDL_GetWindowSizeInPixels() returns the size of the back buffer for a window

Topic: High DPI Support

- SDL 2D Render API has convenience functions:
  - SDL_SetRenderLogicalPresentation() sets a logical size for the content, rendering to an offscreen texture and then scaling it as needed for presentation
  - SDL_RenderCoordinatesToWindow() and SDL_RenderCoordinatesFromWindow() convert between screen coordinates and coordinates in the render viewport
  - SDL_ConvertEventToRenderCoordinates() will convert all coordinates in an event into coordinates in the render viewport

Topic: High DPI Support

- High DPI events:
  - SDL_EVENT_WINDOW_RESIZED is sent when a window is resized in screen coordinates
  - SDL_EVENT_WINDOW_PIXEL_SIZE_CHANGED is sent when the size of a window's backbuffer has changed, which can happen when moving between high and low density displays
  - SDL_EVENT_DISPLAY_SCALE_CHANGED is sent when a display changes scale, which also triggers window pixel size events

Conclusion: High DPI Support

- SDL gives you a consistent way of handling high DPI scenarios across all operating systems
- Your game should handle floating point mouse coordinates, and window sizes that are different than your back buffer sizes
- When you get a pixel size changed event, just rebuild your back buffer

GDC | March 20-24, 2023
San Francisco, CA

Topic: SDL 3.0 GPU API

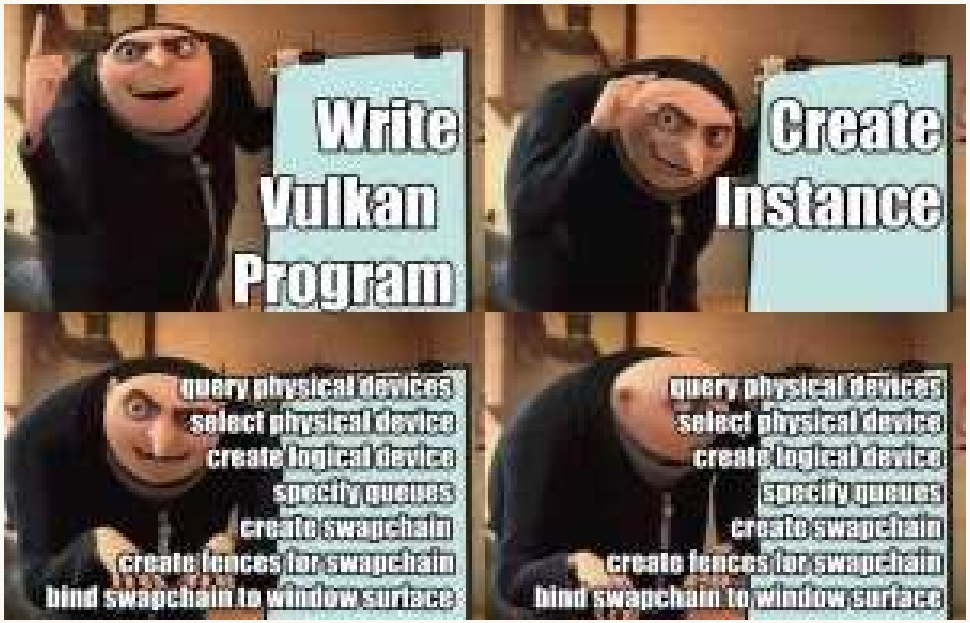
#GDC23

The 2D problem

- SDL's 2D API is limited
- Extending it is hard
- Even in 2D, the thing people want is shaders

The 3D problem

- Most of it is hard to use
- Nothing is portable



SDL 3.0 GPU API

- Turbo charge SDL's rendering offerings
- Offers SDL's focus on simplicity and portability
- Built on next-gen APIs
- Not required!
- Existing 2D API built on top of it

SDL 3.0 GPU Overview

- C-callable API offering modern GPU concepts
- Command queues, pipelines, PSOs, shaders, fences
- Thread safe!
- Uses own shader language and bytecode format

Wait, what?

- Existing offerings didn't meet portability and simplicity goals
- You can use other rendering APIs with SDL if you like!
- Transpiling is possible

SDL 3.0 GPU Shaders

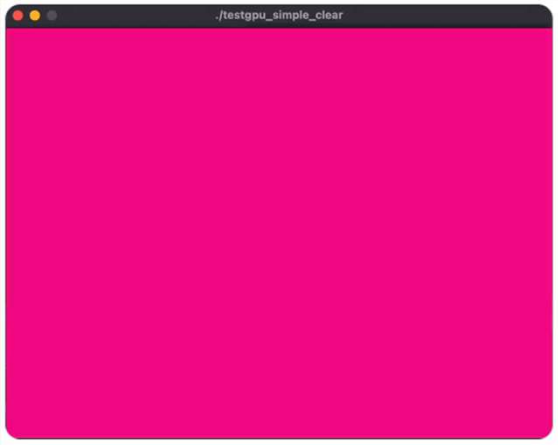
- Uses own shader language and compiled shader bytecode format
- Bytecode is cross-platform: build once, ship everywhere
- Shader compiler is small open source C library
- Use command line to compile, or embed in your own tools or your own game, to use shader source at runtime!

SDL 3.0 GPU Shading Language

- Looks a lot like GLSL, HLSL, MSL, or C
- Removed some footguns
- Improved some syntax in mostly optional ways
- If you've ever written a shader, you can handle this
- Vertex and pixel shaders for now, more to come

SDL 3.0 GPU Example

- The "S" stands for "Simple“
- Vulkan: 1157 lines of C.
- SDL GPU API: 153 lines (including error checking!)



SDL 3.0 GPU Conclusion

- Still early in development, but looking good
- Feedback welcome!

Questions?

- Presented by Sam Lantinga (slouken@libsdl.org) and Ryan Gordon (icculus@icculus.org)
- Slides and code examples at https://www.libsdl.org/gdc2023
- Thank you!