A library for writing game engines

http://www.libsdl.org/

The Name

Simple

DirectMedia

Layer

History

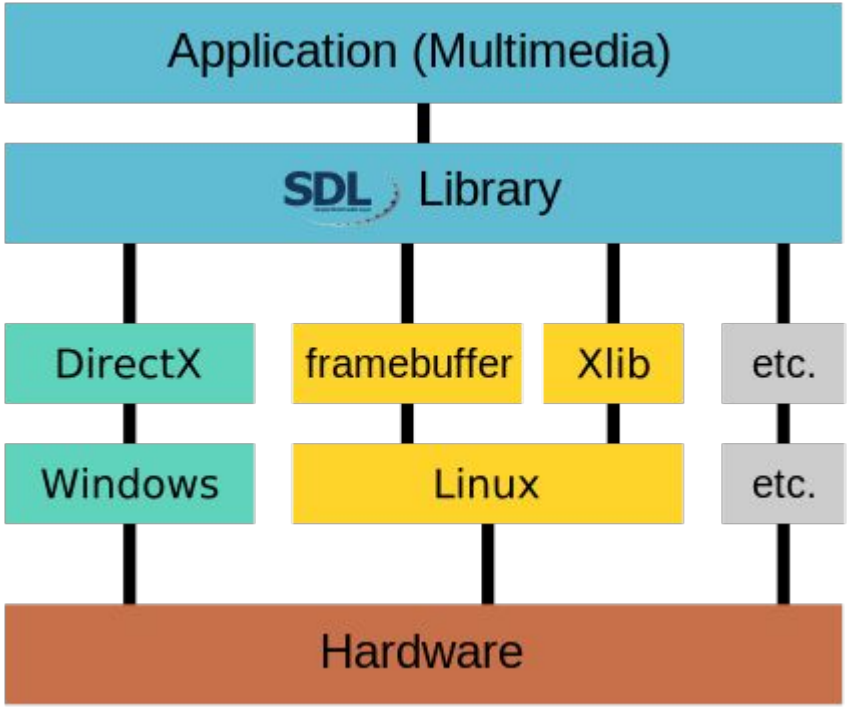
Author: **Sam Lantinga**

- Was lead software engineer at **Blizzard.**
- Now works at **Valve.**

SDL originally released in 1998.

Version 2 released in 2012.

Cross-Platform



http://en.wikipedia.org/wiki/Simple_DirectMedia_Layer

Who uses SDL?

Amnesia	Don’t Starve
World of Goo	Counter-Strike
Half-Life	Portal
Team Fortress 2	Fez
Visual Boy Advance	ZSNES
Trine	Psychonauts

http://en.wikipedia.org/wiki/List_of_games_using_SDL

Features

- 2D graphics
- 3D graphics (with OpenGL or DirectX)
- Window management
- Event handling
- Image loading
- Networking
- Audio
- Font rendering
- Controller support
- Text rendering
- … and more

Design

Core:

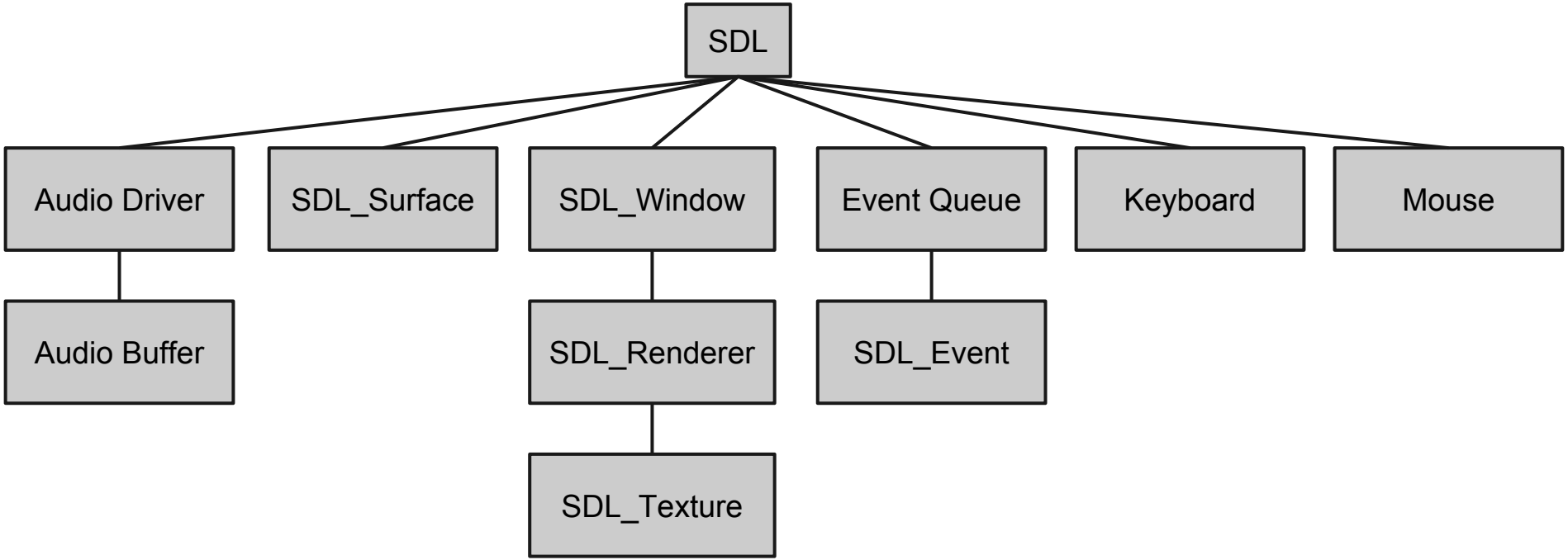
SDL

Video, Event handling,
Audio, Controllers, ...

Add-ons:

SDL_image	SDL_mixer	SDL_net	SDL_ttf	SDL_rtf

Some data structures



```
Initialize SDL, window, renderer;
Load textures;
while (IsGameRunning)
    Poll events;
    Update state of game entities;
    Clear the screen;
    Render everything;
    Flip the screen;
    Sleep until the next frame;
Quit SDL;
```

Typical SDL Game loop

```
Uint8 flag1 = 0x01;
Uint8 flag2 = 0x02;
Uint8 flag3 = 0x04;

Uint8 combinedFlags = flag1 | flag2;

if (combinedFlags & flag1) {
    // do something since flag1 is set
}

if (combinedFlags & flag3) {
    // do something since flag3 is set
}
```

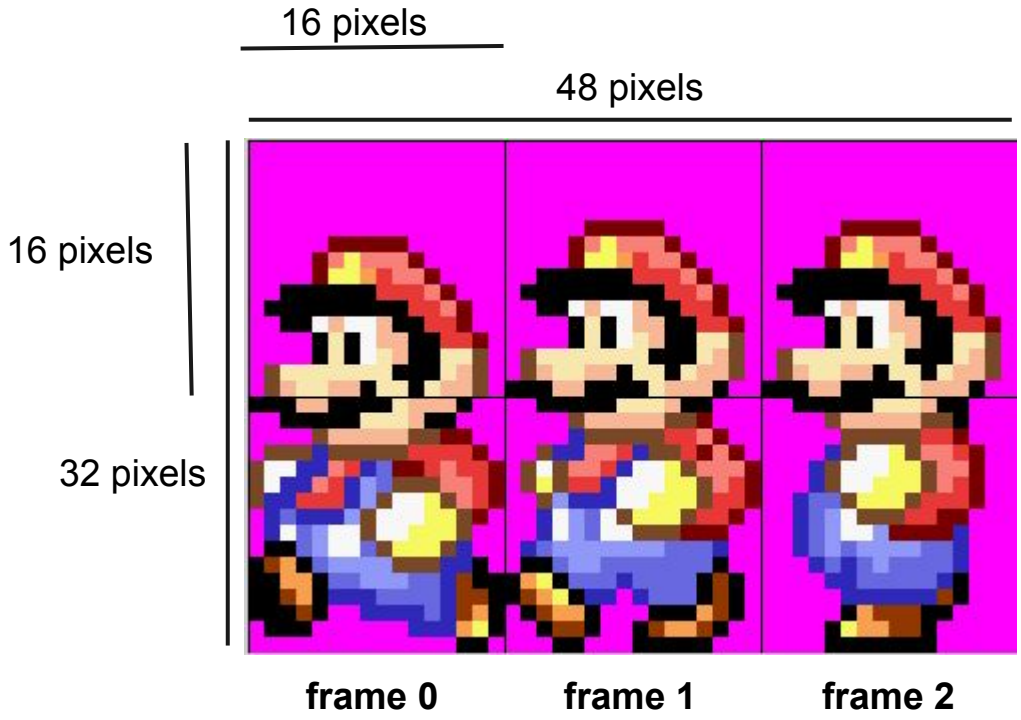
```
flag1 = 0000 0001
flag2 = 0000 0010
flag3 = 0000 0100

  0000 0001 (flag1)
| 0000 0010 (flag2)
  ---------
= 0000 0011 = combinedFlags

  0000 0011 (combinedFlags)
& 0000 0001 (flag1)
  ---------
= 0000 0001 (combinedFlags & flag1)

  0000 0011 (combinedFlags)
& 0000 0100 (flag3)
  ---------
= 0000 0000 (combinedFlags & flag3)
```

Using bit flags for options

```
SDL_Init(SDL_INIT_EVERYTHING);

SDL_Window *window = SDL_CreateWindow(
                         “My Game”, // title of window
                         SDL_WINDOWPOS_UNDEFINED, // x position of window
                         SDL_WINDOWPOS_UNDEFINED, // y position of window
                         640, 480, // dimensions of window
                         0); // window option flags

SDL_Renderer *renderer = SDL_CreateRenderer(
                             window, // window to render on
                             -1, // what driver to use (-1: choose for me)
                             // renderer option flags
                             SDL_RENDERER_ACCELERATED | SDL_RENDERER_PRESENTVSYNC);
```

```
SDL_Window *window;
SDL_Renderer *renderer;

// the lazy way
SDL_CreateWindowAndRenderer(640, 480, 0, &window, &renderer);
```

Initializing SDL



Color Key

www.nes-snes-sprites.com/SuperMarioAllStarsSMB2.html

Sprite sheet aka texture atlas

SDL_PIXELFORMAT_RGB332

SDL_PIXELFORMAT_RGB444

SDL_PIXELFORMAT_RGB555

SDL_PIXELFORMAT_BGR555

SDL_PIXELFORMAT_ARGB4444

SDL_PIXELFORMAT_RGBA4444

SDL_PIXELFORMAT_ABGR4444

SDL_PIXELFORMAT_BGRA4444

SDL_PIXELFORMAT_ARGB1555

SDL_PIXELFORMAT_RGBA5551

SDL_PIXELFORMAT_ABGR1555

SDL_PIXELFORMAT_BGRA5551

SDL_PIXELFORMAT_RGB565

SDL_PIXELFORMAT_BGR565

SDL_PIXELFORMAT_RGB24

SDL_PIXELFORMAT_BGR24

SDL_PIXELFORMAT_RGB888

SDL_PIXELFORMAT_RGBX8888

SDL_PIXELFORMAT_BGR888

SDL_PIXELFORMAT_BGRX8888

SDL_PIXELFORMAT_ARGB8888

SDL_PIXELFORMAT_RGBA8888

SDL_PIXELFORMAT_ABGR8888

SDL_PIXELFORMAT_BGRA8888

Pixel formats. There’s many. Be aware they exist.

Surfaces vs Textures

SDL_Surface (CPU)

- Create with **SDL_LoadBMP.**
- Load more file formats with **IMG_Load.**
- Surfaces are just an array of pixels.
- Pixel format of a surface may not be the same as the pixel format of the window. Convert with **SDL_ConvertSurfaceFormat**.
- Convert to **SDL_Texture** using **SDL_CreateTextureFromSurface**
- Appropriate for a paint program.

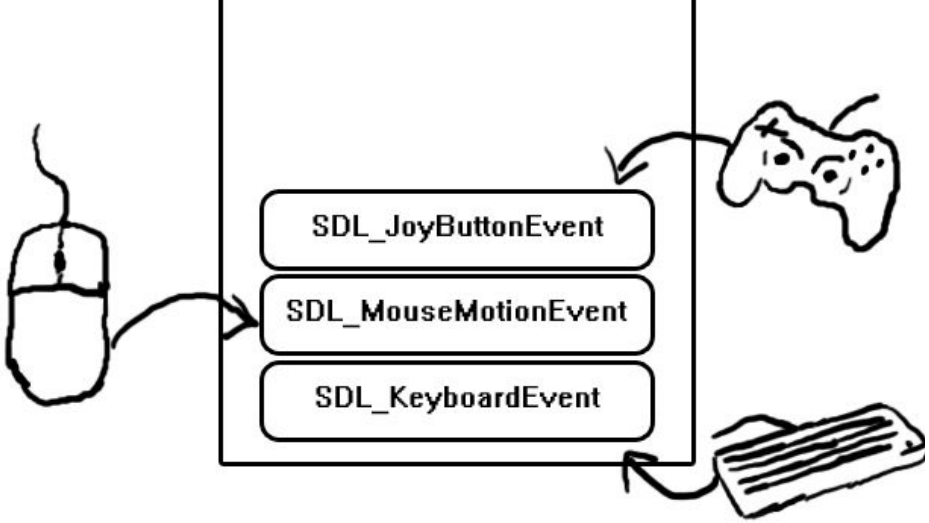
SDL_Texture (GPU)

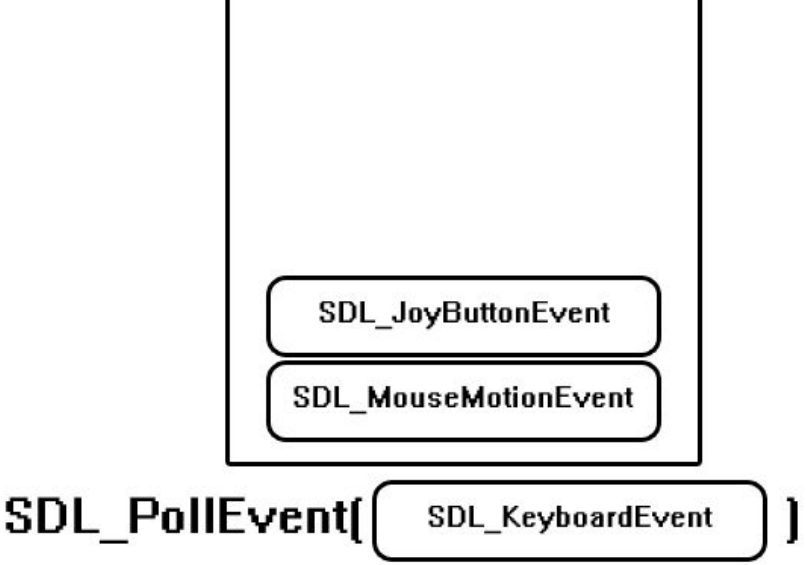
- Stored in the GPU.
- Fast for rendering.
- Can render to a texture. (Instead of the screen.)
- Can render with `SDL_RenderCopy` or `SDL_RenderCopyEx`.
- Can use with OpenGL with `SDL_GL_BindTexture`.
- Appropriate for real-time graphics.

```
// load an image, then convert it to the same format as the window
SDL_Surface *tempSurface = SDL_LoadBMP("path/to/image.bmp");
SDL_Surface *surface = SDL_ConvertSurfaceFormat(tempSurface, SDL_GetWindowPixelFormat(window), 0);
SDL_FreeSurface(tempSurface);

// set the transparent color to hot pink (r = 255, g = 0, b = 255)
SDL_SetColorKey(surface, SDL_TRUE, SDL_MapRGB(surface->format, 255, 0, 255));

// make a texture out of it (this will handle the color key correctly)
SDL_Texture *texture = SDL_CreateTextureFromSurface(renderer, surface);

// no longer need the surface, since we have the texture.
SDL_FreeSurface(surface);
```

```
// pngs can store transparency naturally. IMG_Load takes care of using the right format.
SDL_Surface *surface = IMG_Load("path/to/image.png");
SDL_Texture *texture = SDL_CreateTextureFromSurface(renderer, surface);
SDL_FreeSurface(surface);
```

Loading a texture

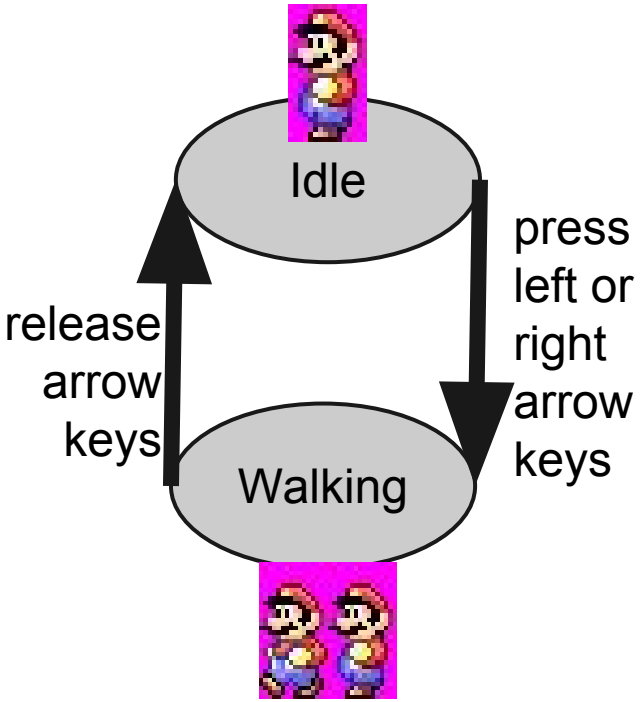




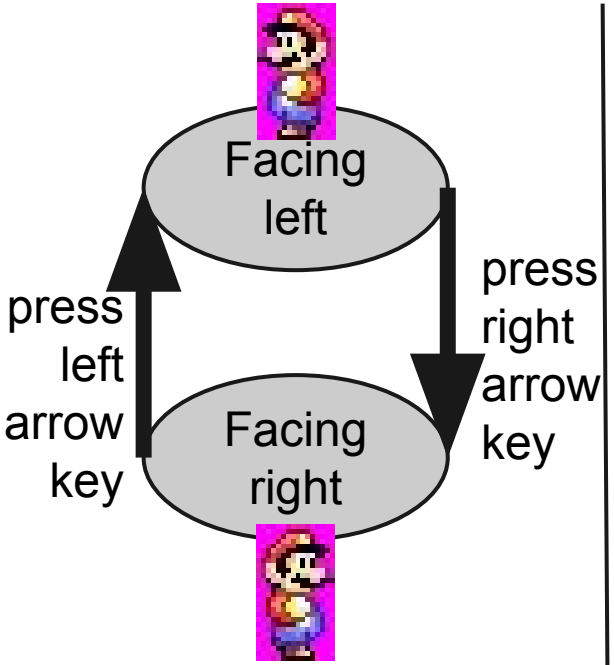

```
SDL_Event e;
while (SDL_PollEvent(&e)) {
    if (e.type == SDL_KEYDOWN) {
        printf("Pressed key: %s\n", SDL_GetKeyName(e.key.keysym.sym));
    }
}
```

http://lazyfoo.net/tutorials/SDL/03_event_driven_programming/index.php

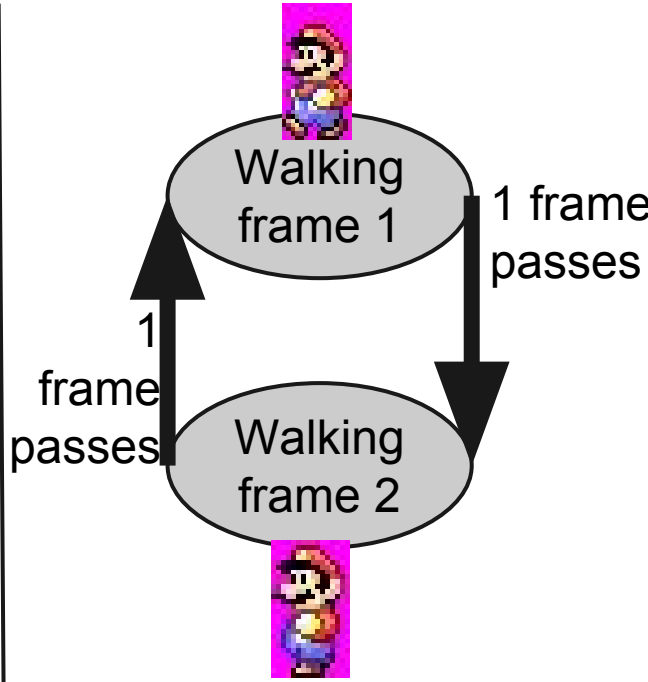
Flushing the event queue



What is mario doing?

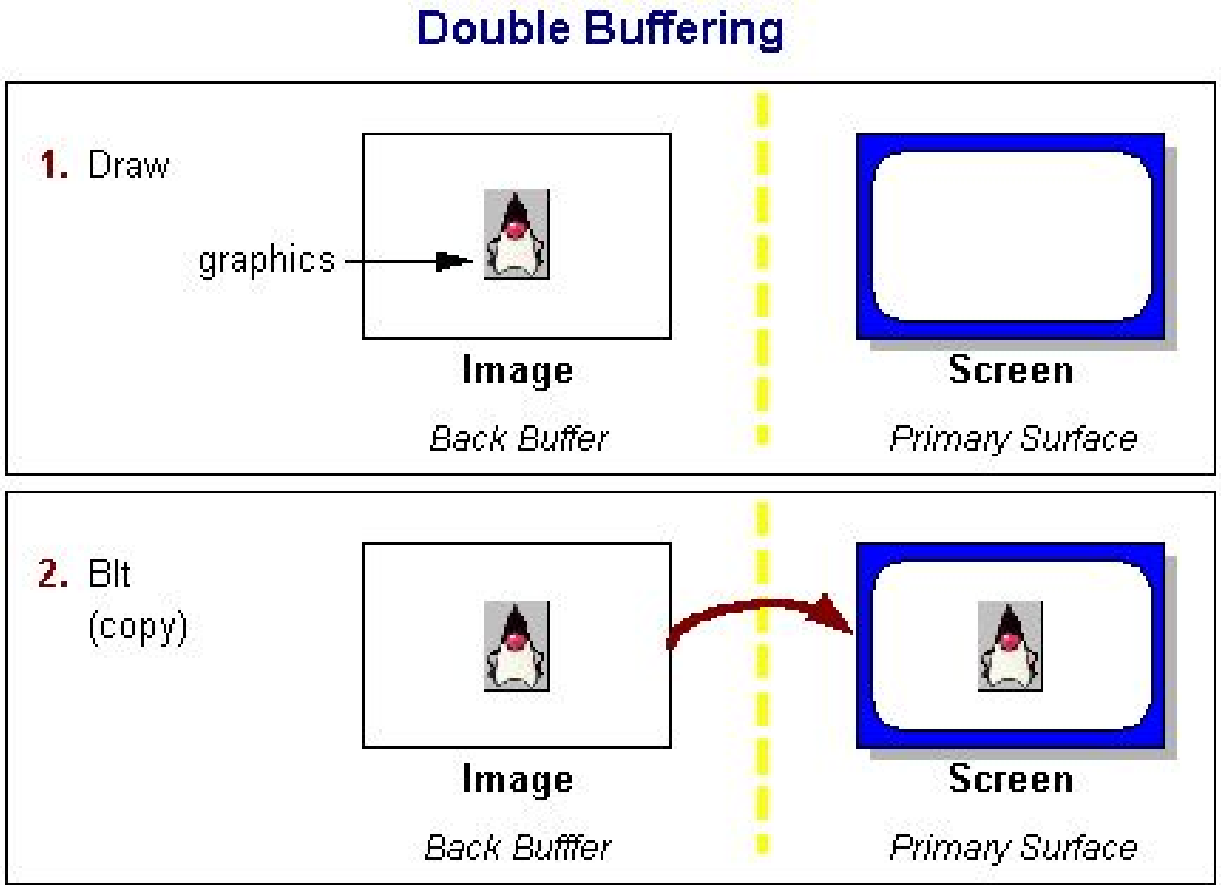


Which direction is he facing?



In what frame of animation is he in?

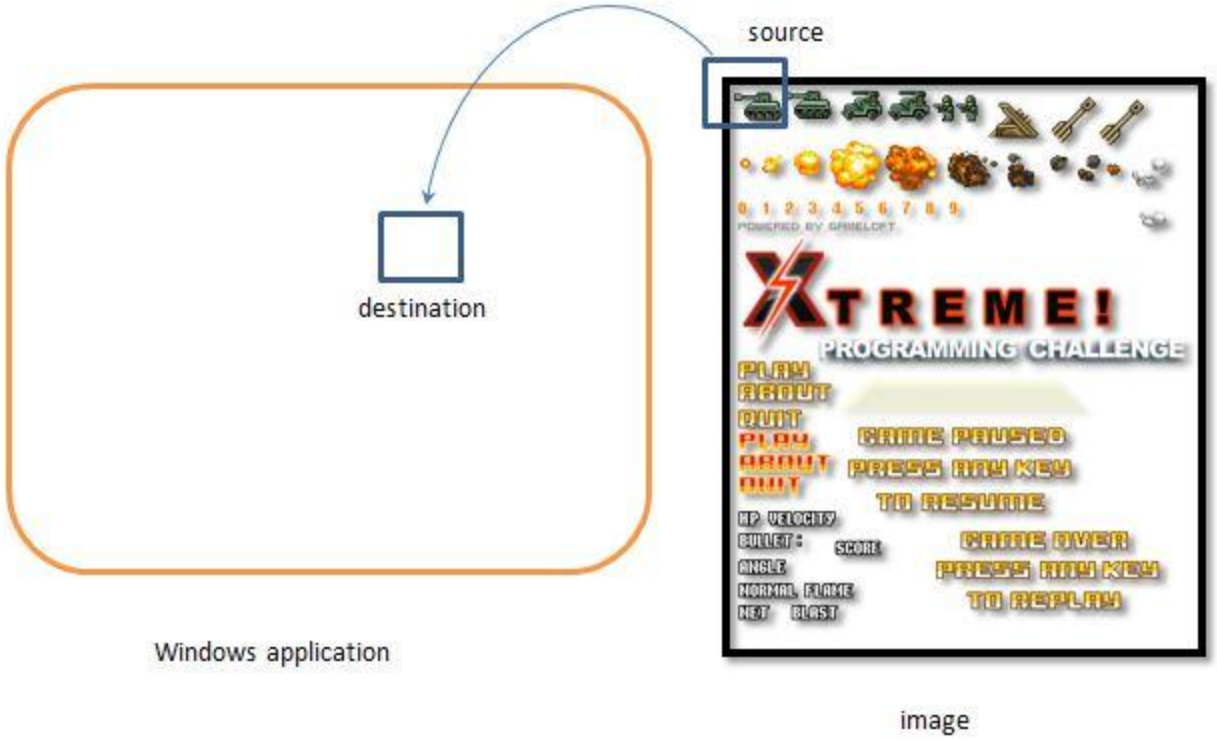
State machines




```
while (true) {
   Clear screen;
   Draw sprite;
   Swap Buffers;
}
```

http://docs.oracle.com/javase/tutorial/extra/fullscreen/doublebuf.html

Double buffering

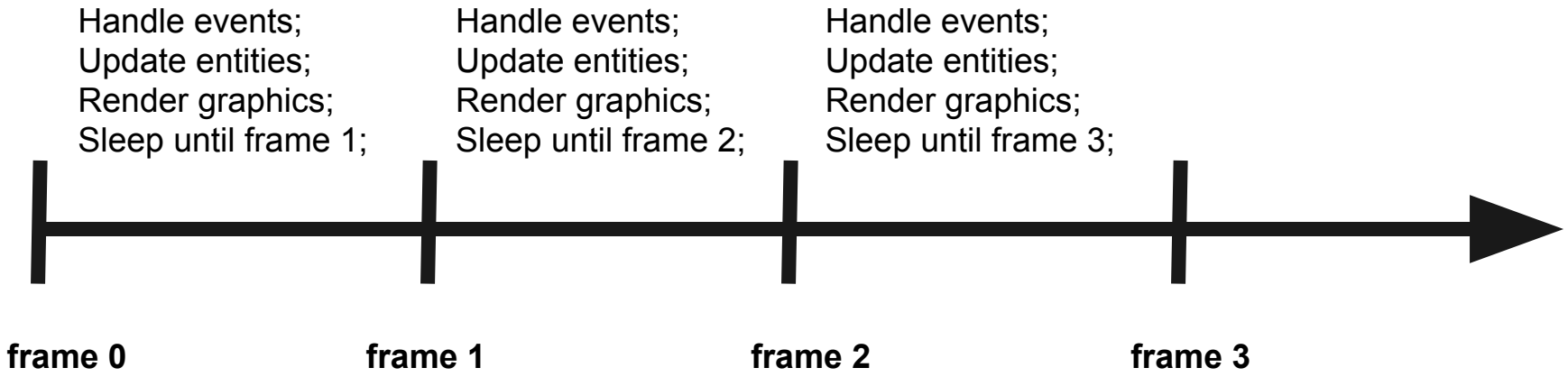



```
SDL_Rect src = {
    what to draw
};

SDL_Rect dst = {
    where to draw
};

SDL_RenderCopy(
    renderer,
    texture,
    &src,
    &dst);
```

http://www.threelas.com/2011/08/basic-using-keyboard-to-control-event.html

Source rectangle / destination rectangle

Frame rate



1000/60 = 16.67 milliseconds

```
SDL_Delay(1000/60); // is this OK?
```

```
Uint32 then = SDL_GetTicks();
while (true) {
    Uint32 now = SDL_GetTicks();
    Uint32 deltaTime = now - then;
    then = now;

    position += velocity * (deltaTime / 1000.0f);

    if (deltaTime < 1000/60) {
        SDL_Delay(1000/60 - deltaTime);
    }
}
```

Calculating delta time

Educate yourself!

Read The Fucking Manual

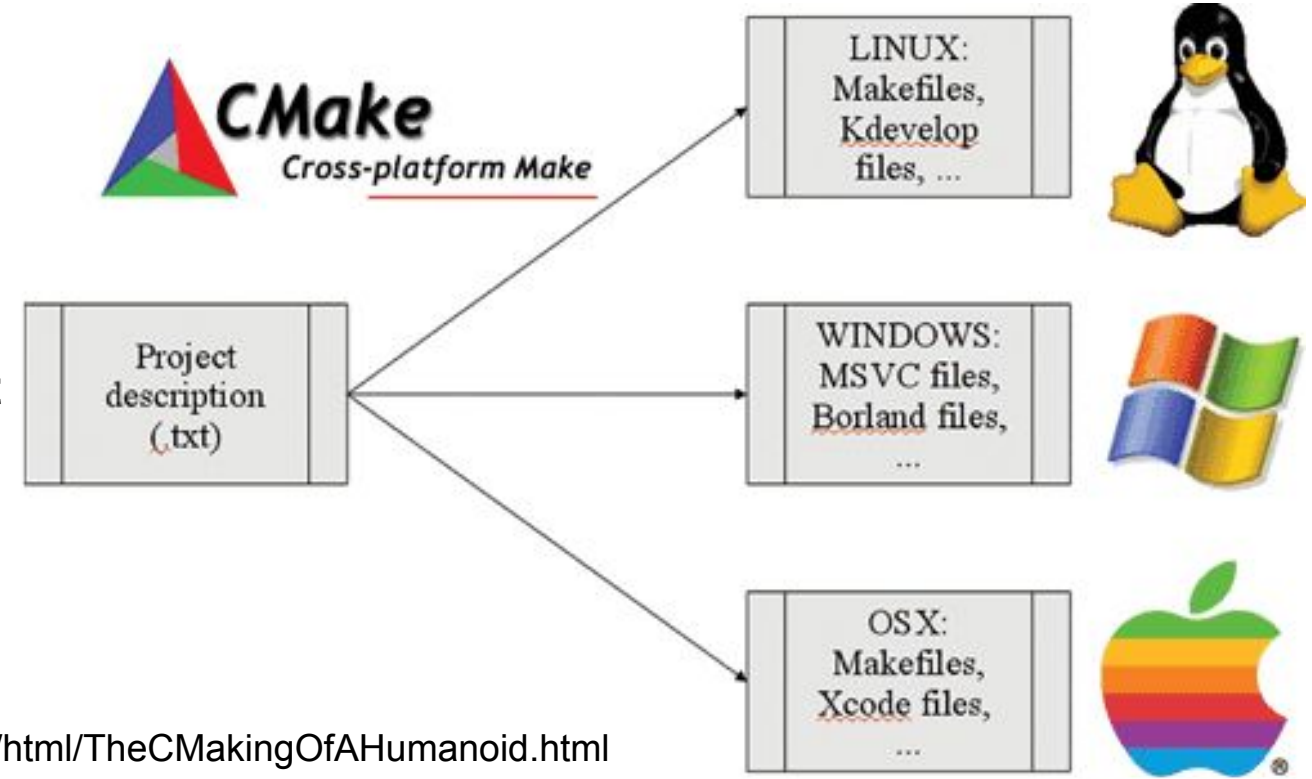
https://wiki.libsdl.org/

“Read The Fucking Source Code”
- Linus Torvalds

http://www.libsdl.org/download-2.0.php

A word on cmake

CMakeLists.txt



http://www.kitware.com/media/html/TheCMakingOfAHumanoid.html

Getting Started (command line/Linux/make)

```
$ git clone https://github.com/UVicGameDev/sdl2-template.git
$ mkdir sdl2-template-build
$ cd sdl2-template-build
$ cmake ../sdl2-template
$ cd template
$ make
$ ./template
```

Getting Started (cmake-gui/Windows/Visual Studio)

(assuming the repository from the last slide was cloned as “sdl”)

* Set the source/build paths as in the image
(the directories leading up to sdl are unimportant)

* Click Configure, choose Visual Studio

* Uncheck DIRECTX and Configure again
(unless you really want to use DIRECTX)

* Click Generate

* Open the VS solution and build it.