Programming

Gameplay, Networking, Graphics, Engine, Tools, UI, Physics, Audio

Programmer

Write Code, Debug code/builds, Optimize code, Refactor code

Critical thinking

Problem solving:

Appreciation method – understand the problem

5 Whys technique – defines the problem

Reframing – looks at the problem from a new perspective

Guess and Check, Find a Pattern, Divide and Conquer – find solutions

Decide on a solution

Logical thinking:

Deductive reasoning – use hypotheses that hold true to draw logical conclusions

Inductive reasoning – uses specific cases to draw general conclusions

De Morgan’s Law !(A && B) == (!A || !B), !(A || B) == (!A && !B)

Commutative Law, Associative Law,

Identity Law A || false == A, A && true == A

Dominance Law A || true == true, A&& false == false

Distributive Law

A && (B || C) == A && B || A && C, A || (B && C) == (A || B) && (A || C)

SDL

int SDL\_Init(Uint32 flags)

flags – SDL\_INIT\_\*: TIMER, AUDIO, VIDEO, JOYSTICK, HAPTIC, GAMECONTROLLER, EVENTS, EVERYTHING

SDL\_Window\* SDL\_CreateWindow(const char\* title, int x, int y, int w, int h, Uint32 flags)

flags – SDL\_WINDOW\_\*: FULLSCREEN, FULLSCREEN\_DESKTOP, OPENGL, HIDDEN, BORDERLESS, RESIZABLE, MINIMIZED, MAXIMIZED, INPUT\_GRABBED, ALLOW\_HIGHDPI

SDL\_Renderer\* SDL\_CreateRenderer(SDL\_Window\* window, int index, Uint32 flags)

index – the index of the rendering driver to initialize, or -1 to initialize the first one supporting the requested flags

flags – SDL\_RENDERER\_\*: SOFTWARE, ACCELERATED, PRESENTVSYNC, TARGETTEXTURE

int SDL\_PollEvent(SDL\_Event\* event)

union SDL\_Event

struct SDL\_KeyboardEvent

Uint32 type – SDL\_KEYDOWN, SDL\_KEYUP

SDL\_Keysym keysym

struct SDL\_Keysym

SDL\_Scancode scancode

SDL\_Keycode sym

const Uint8\* SDL\_GetKeyboardState(int\* numkeys)

numkeys – if non-NULL, receives the length of the returned array

Uint32 SDL\_GetTicks()

return – returns an unsigned 32-bit value representing the number of milliseconds since the SDL library initialized

const char\* SDL\_GetError()

int SDL\_SetError(const char\* fmt, ...)

fmt – a printf() style message format string

… – additional parameters matching % tokens in the fmt string, if any

void SDL\_ClearError()

int SDL\_SetRenderDrawColor(SDL\_Renderer\* renderer, Uint8 r, Uint8 g, Uint8 b, Uint8 a)

int SDL\_SetRenderDrawBlendMode(SDL\_Renderer\* renderer, SDL\_BlendMode blendMode)

blendMode –

|  |  |
| --- | --- |
| SDL\_BLENDMODE\_NONE | no blending  dstRGBA = srcRGBA |
| SDL\_BLENDMODE\_BLEND | alpha blending  dstRGB = (srcRGB \* srcA) + (dstRGB \* (1-srcA))  dstA = srcA + (dstA \* (1-srcA)) |
| SDL\_BLENDMODE\_ADD | additive blending  dstRGB = (srcRGB \* srcA) + dstRGB  dstA = dstA |
| SDL\_BLENDMODE\_MOD | color modulate  dstRGB = srcRGB \* dstRGB  dstA = dstA |

int SDL\_RenderDrawLine(SDL\_Renderer\* renderer, int x1, int y1, int x2, int y2)

int SDL\_RenderDrawLines(SDL\_Renderer\* renderer, const SDL\_Point\* points, int count)

int SDL\_RenderDrawPoint(SDL\_Renderer\* renderer, int x, int y)

int SDL\_RenderDrawPoints(SDL\_Renderer\* renderer, const SDL\_Point\* points, int count)

int SDL\_RenderDrawRect(SDL\_Renderer\* renderer, const SDL\_Rect\* rect)

int SDL\_RenderDrawRects(SDL\_Renderer\* renderer, const SDL\_Rect\* rects, int count)

int SDL\_RenderClear(SDL\_Renderer\* renderer)

int SDL\_RenderFillRect(SDL\_Renderer\* renderer, const SDL\_Rect\* rect)

void SDL\_RenderPresent (SDL\_Renderer \*renderer)

void SDL\_Quit()

IMAGE/TEXTURE

int IMG\_Init(int flags)

flags – IMG\_INIT\_PNG, IMG\_INIT\_JPG, IMG\_INIT\_TIF

return – a bitmask of all the currently initted image loaders.

SDL\_Texture\* IMG\_LoadTexture(SDL\_Renderer \*renderer, const char \*file)

int SDL\_RenderCopy(SDL\_Renderer\* renderer, SDL\_Texture\* texture, const SDL\_Rect\* srcrect, const SDL\_Rect\* dstrect)

int SDL\_RenderCopyEx(SDL\_Renderer\* renderer, SDL\_Texture\* texture, const SDL\_Rect\* srcrect, const SDL\_Rect\* dstrect, const double angle, const SDL\_Point\* center, const SDL\_RendererFlip flip)

void IMG\_Quit()

COLLISION

SDL\_bool SDL\_HasIntersection (const SDL\_Rect \*A, SDL\_Rect \*B)

SDL\_bool SDL\_IntersectRect (const SDL\_Rect\* A, SDL\_Rect \*B, SDL\_Rect \* result)

return – the intersection rectangle of rect A and B

AUDIO

struct Mix\_Chunk

int allocated – 0 memory not allocated, 1 memory allocated

Uint8 \*abuf

Uint32 alen – length of abuf in bytes

Uint8 volume – 0 to 128

int Mix\_Init(int flags)

flags – MIX\_INIT\_MP3

int Mix\_OpenAudio (int frequency, Uint16 format, int channels, int chunksize)

frequency – MIX\_DEFAULT\_FREQUENCY(22050)

format – AUDIO\_S16SYS (good enough)

channels – 2 for stereo, 1 for mono

chunksize – 8192

int Mix\_AllocateChannels (int numchans)

numchans – 8 mixing channels by default

Mix\_Chunk \*Mix\_LoadWAV (const char \* file)

Mix\_Music \*Mix\_LoadMUS (const char \*file)

int Mix\_Volume (int channel, int volume)

volume – from 0 to MIX\_MAX\_VOLUME (128); if > 128, set 128; if < 0, will not set.

int Mix\_VolumeMusic(int volume)

int Mix\_VolumeChunk (Mix\_Chunk \*chunk, int volume)

int Mix\_SetPanning (int channel, Uint8 left, Uint8 right)

channel – channel number to apply effect, MIX\_CHANNEL\_POST for postmix stream

left – volume for the left stereo channel, ranging from 0 to 255. Only works on stereo.

right – the same as left, usually equals to 254 - left for true panning.

int Mix\_PlayMusic (Mix\_Music \*music, int loops)

loops – 0 zero times, -1 loop foever

return – 0 success, -1 errors

int Mix\_PlayChannel (int channel, Mix\_Chunk \*chunk, int loops)

channel – -1 for the first free unreserved channel

chunk – sound effect to play

loops – -1 infinite loops, 0 for playing one time

int Mix\_Playing (int channel)

channel – test if it’s playing, paused is counted as playing, -1 for num of playing channels

int Mix\_HaltChannel (int channel)

channel – channel to stop playing not just pause it, -1 for all channels

CLEANING UP

void Mix\_FreeChunk(Mix\_Chunk\*)

void Mix\_FreeMusic(Mix\_Music\*)

void Mix\_CloseAudio()

void Mix\_Quit()