SDL Dokumentation

Dienstag, 26. April 2016 22:30

Includierungen:

<SDL2/SDL.h> SDL
<SDL_image.h> SDL_Image
<SDL_ttf.h> TTF Text
<SDL_mixer.h>

Offizielle Dokumentation zum Nachschlagen

SDL Initialisierung und Beendigung

SDL_Init(Uint32 flag);

SDL_INIT_TIMER	timer subsystem
SDL_INIT_AUDIO	audio subsystem
SDL_INIT_VIDEO	video subsystem. Automatically initializes the SDL_INIT_EVENTS subsystem
SDL_INIT_JOYSTICK	joystick subsystem
SDL_INIT_HAPTIC	haptic (force feedback) subsystem
SDL_INIT_GAMECONTROLLER	controller subsystem. Automatically initializes the SDL_INIT_JOYSTICK subsystem
SDL_INIT_EVENTS	events subsystem
SDL_INIT_EVERYTHING	all of the above subsystems
SDL_INIT_NOPARACHUTE	compatibility; this flag is ignored

SDL_Quit();

Window und Surface erstellen und Löschen

Window erstellen
SDL_Window* window = NULL;
Surface erstellen
SDL_Surface* surface = NULL;

Window löschen
SDL_DestroyWindow(window);
Surface löschen
SDL_FreeSurface(surface);

Window intialisieren
Window = SDL_CreateWindow("SDL Tutorial", SDL_WINDOWPOS_UNDEFINED,
SDL_WINDOWPOS_UNDEFINED, SCREEN_WIDTH, SCREEN_HEIGHT,
SDL_WINDOW_SHOWN);

SDL_CreateWindow Parameter sind:
1. Name des Windows

2&3. X,Y Koordinaten zum Bildschirm,
4.Flags

SDL_WINDOW_FULLSCREEN	fullscreen window
SDL_WINDOW_FULLSCREEN_DESKTOP	fullscreen window at the current desktop resolution
SDL_WINDOW_OPENGL	window usable with OpenGL context
SDL_WINDOW_SHOWN	window is visible
SDL_WINDOW_HIDDEN	window is not visible
SDL_WINDOW_BORDERLESS	no window decoration
SDL_WINDOW_RESIZABLE	window can be resized
SDL_WINDOW_MINIMIZED	window is minimized
SDL_WINDOW_MAXIMIZED	window is maximized
SDL_WINDOW_INPUT_GRABBED	window has grabbed input focus
SDL_WINDOW_INPUT_FOCUS	window has input focus
SDL_WINDOW_MOUSE_FOCUS	window has mouse focus
SDL_WINDOW_FOREIGN	window not created by SDL
SDL_WINDOW_ALLOW_HIGHDPI	window should be created in high-DPI mode if supported (>= SDL 2.0.1)
SDL_WINDOW_MOUSE_CAPTURE	window has mouse captured (unrelated to INPUT_GRABBED, >= SDL 2.0.4)

Es wird nicht direkt auf das Window gerendert, sondern auf das Surface des Windows.

Surface zum Surface des Windows intialisieren
surface = SDL_GetWindowSurface (window);

surface ist nun das Surface des Windows.

Normales Surface Initialisieren

SDL_Surface* HelloWorld = SDL_LoadBMP("HelloWorld.bmp");

Als PNG Laden

SDL_image.h muss inkludiert und gelinkt werden

IMG_Init(IMG_INIT_PNG); zum Beenden muss IMG_Quit(); ausgeführt werden.

SDL_Surface* HelloWorld = IMG_Load("HelloWorld.png");

Surface auf dem Window darstellen

Da nicht direkt auf das Window sondern auf das Surface des Windows gerendert wird, muss ein anderes Surface auf das Window Surface gerendert werden.

SDL_BlitSurface (HelloWorld, NULL, surface, NULL);

Danach muss das Window noch aktualisiert werden

SDL_UpdateWindowSurface(window);

Parameter sind:

1. Quellsurface
2. SDL_Rect mit Struktur des zu kopierenden Quellsurfaces oder NULL, um das komplette

Quellsurface zu kopieren
3. Zielsurface
4. SDL_Rect mit Struktur des Zielsurfaces oder NULL, für das Komplette Zielsurface

Surface in das gleiche Format wie das Window konvertieren, für bessere Performance

SDL_Surface* betterhelloworld = SDL_ConvertSurface(helloworld, windowSurface-> format, 0);

Parameter:

1. Source Surface
2. Surface des Windows->format
3. 0, NULL

Surface auf anderes Surface stretchen

```
SDL_Rect stretchRect;
 stretchRect.x = 0;
 stretchRect.y = 0;
 stretchRect.w = SCREEN_WIDTH;
 stretchRect.h = SCREEN_HEIGHT;

SDL_BlitScaled( gStretchedSurface, NULL, gScreenSurface, &stretchRect );
```

Parameter:

1. Das Zielsurface
2. NULL
3. Das Quellensurface, das gestreched wird
4. Adresse eines SDL_Rect mit den neuen Werten

Surface speicher löschen

SDL_FreeSurface(SDL_Surface*);

Renderer und Textures

Renderer Deklarieren und Definieren für Geometrie oder anderes

SDL_Renderer* Renderer = SDL_CreateRenderer (Window, -1, SDL_RENDERER_ACCELERATED);

Parameter:

1. SDL_Window* auf das gerendert werden soll
2. Index des Render Treibers oder -1 für den ersten, der unterstütz wird (kommt auf die flags an)
3. Flags

SDL_RENDERER_SOFTWARE	the renderer is a software fallback
SDL_RENDERER_ACCELERATED	the renderer uses hardware acceleration

SDL_RENDERER_SOFTWARE	the renderer is a software fallback
SDL_RENDERER_ACCELERATED	the renderer uses hardware acceleration
SDL_RENDERER_PRESENTVSYNC	present is synchronized with the refresh rate
SDL_RENDERER_TARGETTEXTURE	the renderer supports rendering to texture

Farbe, mit der gerendert wird definieren

SDL_SetRenderDrawColor (Renderer, 0xFF, 0xFF, 0xFF, 0xFF);

Parameter:
1. Der Renderer, der benutzt wird (SDL_Renderer*)
2. 2/3/4 RGB Farben
3. Alpha Wert der benutzt werden soll

Texture Deklarieren und Definieren

SDL_Texture* Texture = SDL_CreateTextureFromSurface (Renderer, Surface); Texture hat nun die Eigenschaften des Surfaces.

Parameter:
1. Renderer, der benutzt werden soll
2. Surface, von welchem die Werte übernommen werden sollen (x,y,w,h usw.)

Window Clearen

SDL_RenderClear(Renderer); cleared Window mit der bei SetRenderDrawColor definierten Farbe.

Textur Rendern

SDL_RenderCopy(Renderer, Texture, NULL, NULL);

Parameter:
1. Renderer, auf den gerendert wird
2. Textur, die gerendert wird
3. SDL_Rect* um zu bestimmen, was von der Textur gerendert wird
4. SDL_Rect* um zu bestimmen, auf welchen Bereich des Renderers gerendert wird

Window, Renderer Updaten

SDL_RenderPresent(gRenderer);

Speicher Löschen:

SDL_DestroyTexture(Texture) ;
SDL_DestroyRenderer(Renderer) ;

Geometrie Rendern

Es wird mit der SDL_SetRenderDrawColor definierten Farbe gerendert

Gefülltes Rechteck

Rect für Position und Größe des Rechtecks

SDL_Rect fillRect = { SCREEN_WIDTH / 4, SCREEN_HEIGHT / 4, SCREEN_WIDTH / 2, SCREEN_HEIGHT / 2 };

Form des Rects rendern

SDL_RenderFillRect(gRenderer, &fillRect);

Nur Umrandung des Rechtecks rendern

SDL_Rect outlineRect {SCREEN_WIDTH / 6, SCREEN_HEIGHT / 6, SCREEN_WIDTH * 2 / 3, SCREEN_HEIGHT * 2 / 3 };

SDL_RenderDrawRect(Renderer, &outlineRect);

Linie Rendern

SDL_RenderDrawLine(Renderer, 0, SCREEN_HEIGHT / 2, SCREEN_WIDTH, SCREEN_HEIGHT / 2);

Punkt Rendern

SDL_RenderDrawPoint(Renderer, SCREEN_WIDTH / 2, SCREEN_HEIGHT / 2);

Events

In SDL, Events dienen dazu Tastatureingaben, Mauseingaben, Joystickeingaben oder ähnliches zu speichern, um auszuwerten.

Mit SDL_PollEvent(SDL_Event* e) kann ein Event abgefrag werden.

Beispiel:

```
Bool quit = false;

SDL_Event e;
While (quit==false){
While ( SDL_PollEvent(&e) != 0 ){

If (e.type == SDL_QUIT){

        Quit = true;

        }


    }

}
```

Tastaturabfragen mit Events

SDL mit ESC beenden

```
Bool quit = false;

SDL_Event e;
While (quit==false){
While ( SDL_PollEvent(&e) != 0 ){

If (e.type == SDL_KEYDOWN){

        Switch (e.key.keysym.sym){
            Case SDLK_ESCAPE: quit = true;


            }

        }


    }

}
```

Infos:
https://wiki.libsdl.org/SDL_KeyboardEvent
https://wiki.libsdl.org/SDL_Keysym
https://wiki.libsdl.org/SDL_Keycode

Zeitfunktionen

Zeit abwarten
SDL_Delay(Uint32 milliseconds);