graphics2d-Kurzreferenz

Callbacks

`on_ready()`
Wird vom Framework aufgerufen, sobald das Framework initialisiert ist und du mit dem Zeichnen usw. beginnen kannst.

`on_draw()`
Wird vom Framework höchstens einmal pro Frame aufgerufen, um den Fensterinhalt neu zu zeichnen. Wenn `ALWAYS_REDRAW` wahr ist, wird `draw()` in jedem Frame aufgerufen, ansonsten führt ein Aufruf von request_redraw() dazu, dass das Framework `draw()` aufruft.

`on_update(dt)`
Wird vom Framework höchstens `MAX_FPS` mal pro Sekunde aufgerufen. Hier kannst du den Zustand deiner Grafik in Abhängigkeit der vergangenen Zeit verändern. Um die Framerate nicht einbrechen zu lassen, solltest du hier (und in allen anderen Callbacks) nichts zun, das mehr Zeit benötigt als `1/MAX_FPS` Sekunden.

`dt:` delta time, die ungefähre Anzahl Millisekunden seit dem letzten Aufruf. Bei 60 FPS beträgt
dieser Wert ungefähr 16.

`on_input(event)`
Wird vom Framework aufgerufen, wenn ein Input des Benutzers vorliegt, z.B. ein Tastendruck, eine Mausbewegung oder ein Gamepad-Ereignis.

`event:` Liefert genauere Informationen zum Event. Sein Typ kann über event.type abgefragt
werden, weitere vorhandene Informationen hängen vom Typ des events ab.

`on_exit()`
Wird vom Framework aufgerufen, bevor `go()` beendet wird.

`on_resized(breite, höhe)`
Wird vom Framework aufgerufen, wenn die Fenstergrösse geändert hat.

`breite:` Die neue Breite des Fensters
höhe: Die neue Höhe des Fensters

Weitere Grafikfunktionen

`go()`:
Konfiguriert das Framework und startet seine Event Loop. `go()` sollte die letzte Anweisung in deinem Programm sein; nachdem `go()` zurückkehrt, führen sämtliche Funktionsaufrufe in das graphics2d-Framework zu einem undefinierten Ergebnis.

set_window_title(title):

Setzt den Titel des Grafikfensters. Hinweis: go() überschreibt diesen Titel, deshalb ist es sinnvoller, den Titel erst im `ready`-Callback zu setzen.

`title:` Der Titel

request_redraw():

Verlangt das Neuzeichnen der Grafik via `draw()` in diesem Frame. Wenn `ALWAYS_REDRAW` wahr ist, ist die Verwendung dieser Funktion nicht nötig, da `draw()` in diesem Fall ohnehin in jedem Frame aufgerufen wird.

draw_line(start, end, color, width=1):

Zeichnet eine Linie von start zu end in der angegebenen Farbe und mit der angegebenen Strichdicke.

`start:` Ein (x, y)-Tupel, Startpunkt der Linie
`end:` Ein (x, y)-Tupel, Endpunkt der Linie
`color:` Ein Color-Objekt, bezeichnet die Farbe der Linie
`width`: Die Breite der Linie

draw_polyline(points, color, is_closed=False, width=1):

Zeichnet eine Linie, welche alle angegebenen Punkte verbindet. Wenn is_closed wahr ist, wird auch der letzte mit dem ersten Punkt verbunden und es entsteht ein Polygon.

`points:` Eine Liste mit (x, y)-Tupeln, welche die Punktkoordinaten angeben
`color:` Ein Color-Objekt
`is_closed:` Wenn True, wird der letzte Punkt mit dem ersten verbunden, sonst nicht.
`width`: Die Breite der Linie

draw_filled_rect(topleft, size, color):

Zeichnet ein mit der Farbe color ausgefülltes Rechteck.

`topleft`: Ein (x,y)-Tupel, das die Position der oberen linken Ecke des Rechtecks angibt
`size`: Ein (breite,höhe)-Tupel, das die Breite und Höhe des Rechtecks angibt.
`color`: Ein Color-Objekt, bezeichnet die Farbe des Rechtecks

draw_circle(center, radius, color, width=1):

Zeichnet einen Kreis.

`center:` Ein (x,y)-Tupel, das die Koordinaten des Kreismittelpunkts angibt
`radius:` Der Radius des Kreises
`color:` Ein Color-Objekt, bezeichnet die Farbe der Kreislinie
`width:` Die Breite der Kreislinie

draw_filled_circle(center, radius, color):

Zeichnet einen mit einer Farbe ausgefüllten Kreis.

`center:` Ein (x,y)-Tupel, das die Koordinaten des Kreismittelpunkts angibt
`radius:` Der Radius des Kreises
`color:` Ein Color-Objekt

draw_text(fontname, fontsize, text, position, color, antialiased=True, background=None):

Rendert Text.

`fontname:` Der Fontname als String.
`fontsize:` Die Schriftgrösse
`text:` Der Text, der gezeichnet werden soll
`position:` Dein (x,y)-Koordinatentupel, welche die Textposition angeben
`color:` Die Farbe des Textes

draw_surface(source_surface, destination_position, source_area=None):

Zeichnet eine Surface (z.B. ein mit load_image geladenes Bild) an die angegebene Position.

`source_surface:` Die Surface, die gezeichnet werden soll
`destination_position:` Ein (x,y)-Tupel, gibt die Koordinaten der linken oberen Ecke an
`source_area:` Ein Rect, das den Bildauschnitt angibt, der aus source_surface gezeichnet wird

get_text_size(fontname, fontsize, text):

Liefert ein (b, h)-Tupel aus Breite und Höhe eines Rechtecks, das den Text umfassen würde.

`fontname:` Der Name des zu verwendenen Fonts (als String)
`fontsize:` Die Grösse des zu verwendenden Fonts
`text:` Der Text, dessen Grösse bestimmt werden soll.

get_window_size():

Liefert die Grösse des Grafikfensters als (breite, höhe)-Tupel zurück.

get_default_fontname():

Liefert den Namen des Default-Fonts zurück.

get_all_fontnames():

Liefert eine Liste aller verfügbaren Fontnamen zurück.

load_image(filename):

Lädt das Bild mit dem angegebenen Dateinamen. Die gängigsten Grafikformate werden unterstützt,
z.B. jpg, png, webp, gif, bmp etc, inklusive einfache SVG-Grafiken.

`filename:` Der Name bzw Pfad zum Bild.

save_screen(filename):

Speichert den Inhalt des Grafikfensters unter dem angegebenen Dateinamen. Achtung: Bereits existierende Dateien werden überschrieben!

`filename:` Der Name bzw Pfad zum Bild.

Konfigurationskonstanten

Diese Konstanten kannst du in deiner Applikation setzen, um das Framework zu konfigurieren, bevor du es mit go() startest. Wenn du sie nicht setzt, verwendet das Framework voreingestellte Werte.

`WIDTH`	Legt die Breite des Grafikfensters fest
`HEIGHT`	Legt die Höhe des Grafikfensters fest
`ALWAYS_REDRAW`	Falls True, wird draw() in jedem Frame aufgerufen, sonst nur nach **`request_redraw()`**
`MAX_FPS`	Begrenzt die Frames pro Sekunde (und updates pro Sekunde)
`FULLSCREEN`	Falls True, startet das Framework im Fullscreen-Modus
`RESIZABLE`	Falls True, kann die Grösse des Grafikfensters vom Benutzer verändert werden.
`DEFAULT_FONT_SIZE`	Legt die Standard-Grösse für Fonts fest.

Vordefinierte Konstanten

`BLACK`	Ein vordefiniertes Color-Objekt für die Farbe Schwarz
`WHITE`	Ein vordefiniertes Color-Objekt für die Farbe Weiss
`RED`	Ein vordefiniertes Color-Objekt für die Farbe Rot
`GREEN`	Ein vordefiniertes Color-Objekt für die Farbe Grün
`BLUE`	Ein vordefiniertes Color-Objekt für die Farbe Blau
`MOUSEMOTION`	Event-Typ für «Mausbewegung»
`MOUSEBUTTONDOWN`	Event-Typ für «Maustaste gedrückt»
`MOUSEBUTTONUP`	Event-Typ für «Maustaste losgelassen»
`KEYDOWN`	Event-Typ für «Taste gedrückt»
`KEYUP`	Event-Typ für «Taste losgelassen»