raylib

A simple and easy-to-use library to enjoy videogames programming

[raylib Discord server][github.com/raysan5/raylib]

raylib v3.0 quick reference card [download as PDF]

module: core

```
// Window-related functions
void InitWindow(int width, int height, const char *title);              // Initialize window and OpenGL context
bool WindowShouldClose(void);                                           // Check if KEY_ESCAPE pressed or Close icon pressed
void CloseWindow(void);                                                 // Close window and unload OpenGL context
bool IsWindowReady(void);                                               // Check if window has been initialized successfully
bool IsWindowMinimized(void);                                           // Check if window has been minimized (or lost focus)
bool IsWindowResized(void);                                             // Check if window has been resized
bool IsWindowHidden(void);                                              // Check if window is currently hidden
bool IsWindowFullscreen(void);                                          // Check if window is currently fullscreen
void ToggleFullscreen(void);                                            // Toggle fullscreen mode (only PLATFORM_DESKTOP)
void UnhideWindow(void);                                                // Show the window
void HideWindow(void);                                                  // Hide the window
void SetWindowIcon(Image image);                                        // Set icon for window (only PLATFORM_DESKTOP)
void SetWindowTitle(const char *title);                                 // Set title for window (only PLATFORM_DESKTOP)
void SetWindowPosition(int x, int y);                                   // Set window position on screen (only PLATFORM_DESKTOP)
void SetWindowMonitor(int monitor);                                     // Set monitor for the current window (fullscreen mode)
void SetWindowMinSize(int width, int height);                           // Set window minimum dimensions (for FLAG_WINDOW_RESIZABLE)
void SetWindowSize(int width, int height);                              // Set window dimensions
void *GetWindowHandle(void);                                            // Get native window handle
int GetScreenWidth(void);                                               // Get current screen width
int GetScreenHeight(void);                                              // Get current screen height
int GetMonitorCount(void);                                              // Get number of connected monitors
int GetMonitorWidth(int monitor);                                       // Get primary monitor width
int GetMonitorHeight(int monitor);                                      // Get primary monitor height
int GetMonitorPhysicalWidth(int monitor);                               // Get primary monitor physical width in millimetres
int GetMonitorPhysicalHeight(int monitor);                              // Get primary monitor physical height in millimetres
Vector2 GetWindowPosition(void);                                        // Get window position XY on monitor
const char *GetMonitorName(int monitor);                                // Get the human-readable, UTF-8 encoded name of the primary monitor
const char *GetClipboardText(void);                                     // Get clipboard text content
void SetClipboardText(const char *text);                                // Set clipboard text content

// Cursor-related functions
void ShowCursor(void);                                                  // Shows cursor
void HideCursor(void);                                                  // Hides cursor
bool IsCursorHidden(void);                                              // Check if cursor is not visible
void EnableCursor(void);                                                // Enables cursor (unlock cursor)
void DisableCursor(void);                                               // Disables cursor (lock cursor)

// Drawing-related functions
void ClearBackground(Color color);                                      // Set background color (framebuffer clear color)
void BeginDrawing(void);                                                // Setup canvas (framebuffer) to start drawing
void EndDrawing(void);                                                  // End canvas drawing and swap buffers (double buffering)
void BeginMode2D(Camera2D camera);                                      // Initialize 2D mode with custom camera (2D)
void EndMode2D(void);                                                   // Ends 2D mode with custom camera
void BeginMode3D(Camera3D camera);                                      // Initializes 3D mode with custom camera (3D)
void EndMode3D(void);                                                   // Ends 3D mode and returns to default 2D orthographic mode
void BeginTextureMode(RenderTexture2D target);                          // Initializes render texture for drawing
void EndTextureMode(void);                                              // Ends drawing to render texture
void BeginScissorMode(int x, int y, int width, int height);             // Begin scissor mode (define screen area for following drawing)
void EndScissorMode(void);                                              // End scissor mode

// Screen-space-related functions
Ray GetMouseRay(Vector2 mousePosition, Camera camera);                  // Returns a ray trace from mouse position
Matrix GetCameraMatrix(Camera camera);                                  // Returns camera transform matrix (view matrix)
Matrix GetCameraMatrix2D(Camera2D camera);                              // Returns camera 2d transform matrix
Vector2 GetWorldToScreen(Vector3 position, Camera camera);              // Returns the screen space position for a 3d world space position
Vector2 GetWorldToScreenEx(Vector3 position, Camera camera,int width, int height); // Returns size position for a 3d world space position
Vector2 GetWorldToScreen2D(Vector2 position, Camera2D camera);          // Returns the screen space position for a 2d camera world space position
Vector2 GetScreenToWorld2D(Vector2 position, Camera2D camera);          // Returns the world space position for a 2d camera screen space position

// Timing-related functions
void SetTargetFPS(int fps);                                             // Set target FPS (maximum)
int GetFPS(void);                                                       // Returns current FPS
float GetFrameTime(void);                                               // Returns time in seconds for last frame drawn
double GetTime(void);                                                   // Returns elapsed time in seconds since InitWindow()

// Color-related functions
int ColorToInt(Color color);                                            // Returns hexadecimal value for a Color
Vector4 ColorNormalize(Color color);                                    // Returns color normalized as float [0..1]
Color ColorFromNormalized(Vector4 normalized);                          // Returns color from normalized values [0..1]
Vector3 ColorToHSV(Color color);                                        // Returns HSV values for a Color
Color ColorFromHSV(Vector3 hsv);                                        // Returns a Color from HSV values
Color GetColor(int hexValue);                                           // Returns a Color struct from hexadecimal value
Color Fade(Color color, float alpha);                                   // Color fade-in or fade-out, alpha goes from 0.0f to 1.0f

// Misc. functions
void SetConfigFlags(unsigned int flags);                                // Setup window configuration flags (view FLAGS)
void SetTraceLogLevel(int logType);                                     // Set the current threshold (minimum) log level
void SetTraceLogExit(int logType);                                      // Set the exit threshold (minimum) log level
void SetTraceLogCallback(TraceLogCallback callback);                    // Set a trace log callback to enable custom logging
void TraceLog(int logType, const char *text, ...);                      // Show trace log messages (LOG_DEBUG, LOG_INFO, LOG_WARNING, LOG_ERROR)
void TakeScreenshot(const char *fileName);                              // Takes a screenshot of current screen (saved a .png)
int GetRandomValue(int min, int max);                                   // Returns a random value between min and max (both included)

// Files management functions
unsigned char *LoadFileData(const char *fileName, int *bytesRead);      // Load file data as byte array (read)
void SaveFileData(const char *fileName, void *data, int bytesToWrite);  // Save data to file from byte array (write)
char *LoadFileText(const char *fileName);                               // Load text data from file (read), returns a '\0' terminated string
void SaveFileText(const char *fileName, char *text);                    // Save text data to file (write), string must be '\0' terminated
bool FileExists(const char *fileName);                                  // Check if file exists
bool IsFileExtension(const char *fileName, const char *ext);            // Check file extension
bool DirectoryExists(const char *dirPath);                              // Check if a directory path exists
const char *GetExtension(const char *fileName);                         // Get pointer to extension for a filename string
const char *GetFileName(const char *filePath);                          // Get pointer to filename for a path string
const char *GetFileNameWithoutExt(const char *filePath);                // Get filename string without extension (uses static string)
const char *GetDirectoryPath(const char *filePath);                     // Get full path for a given fileName with path (uses static string)
const char *GetPrevDirectoryPath(const char *dirPath);                  // Get previous directory path for a given path (uses static string)
const char *GetWorkingDirectory(void);                                  // Get current working directory (uses static string)
char **GetDirectoryFiles(const char *dirPath, int *count);              // Get filenames in a directory path (memory should be freed)
void ClearDirectoryFiles(void);                                         // Clear directory files paths buffers (free memory)
bool ChangeDirectory(const char *dir);                                  // Change working directory, returns true if success
bool IsFileDropped(void);                                               // Check if a file has been dropped into window
char **GetDroppedFiles(int *count);                                     // Get dropped files names (memory should be freed)
void ClearDroppedFiles(void);                                           // Clear dropped files paths buffer (free memory)
long GetFileModTime(const char *fileName);                              // Get file modification time (last write time)

unsigned char *CompressData(unsigned char *data, int dataLength, int *compDataLength);        // Compress data (DEFLATE algorythm)
unsigned char *DecompressData(unsigned char *compData, int compDataLength, int *dataLength);  // Decompress data (DEFLATE algorythm)

// Persistent storage management
int LoadStorageValue(int position);                                     // Load integer value from storage file (from defined position)
void SaveStorageValue(int position, int value);                         // Save integer value to storage file (to defined position)

void OpenURL(const char *url);                                          // Open URL with default system browser (if available)

//------------------------------------------------------------------------------------
// Input Handling Functions
//------------------------------------------------------------------------------------

// Input-related functions: keyb
bool IsKeyPressed(int key);                                             // Detect if a key has been pressed once
bool IsKeyDown(int key);                                                // Detect if a key is being pressed
bool IsKeyReleased(int key);                                            // Detect if a key has been released once
bool IsKeyUp(int key);                                                  // Detect if a key is NOT being pressed
int GetKeyPressed(void);                                                // Get latest key pressed
void SetExitKey(int key);                                               // Set a custom key to exit program (default is ESC)

// Input-related functions: gamepads
bool IsGamepadAvailable(int gamepad);                                   // Detect if a gamepad is available
bool IsGamepadName(int gamepad, const char *name);                      // Check gamepad name (if available)
const char *GetGamepadName(int gamepad);                                // Return gamepad internal name id
bool IsGamepadButtonPressed(int gamepad, int button);                   // Detect if a gamepad button has been pressed once
bool IsGamepadButtonDown(int gamepad, int button);                      // Detect if a gamepad button is being pressed
bool IsGamepadButtonReleased(int gamepad, int button);                  // Detect if a gamepad button has been released once
bool IsGamepadButtonUp(int gamepad, int button);                        // Detect if a gamepad button is NOT being pressed
int GetGamepadButtonPressed(void);                                      // Get the last gamepad button pressed
int GetGamepadAxisCount(int gamepad);                                   // Return gamepad axis count for a gamepad
float GetGamepadAxisMovement(int gamepad, int axis);                    // Return axis movement value for a gamepad axis

// Input-related functions: mouse
bool IsMouseButtonPressed(int button);                                  // Detect if a mouse button has been pressed once
bool IsMouseButtonDown(int button);                                     // Detect if a mouse button is being pressed
bool IsMouseButtonReleased(int button);                                 // Detect if a mouse button has been released once
bool IsMouseButtonUp(int button);                                       // Detect if a mouse button is NOT being pressed
int GetMouseX(void);                                                    // Returns mouse position X
```