**Protheus**

Version: Prototype

Contents

[Overview 3](#_Toc397862992)

[Getting Started 3](#_Toc397862993)

[Graphics 4](#_Toc397862994)

[Sprite Batcher 4](#_Toc397862995)

[Sprite Manager 4](#_Toc397862996)

[Animated Sprite 4](#_Toc397862997)

[Sprite 4](#_Toc397862998)

[Renderer 4](#_Toc397862999)

[Window 4](#_Toc397863000)

[Text Renderer 4](#_Toc397863001)

[Audio 4](#_Toc397863002)

[Scripting 4](#_Toc397863003)

# Overview

Protheus is a 2D game engine utilizing Lua as the scripting extension. Protheus was designed with flexibility in mind.

## Getting Started

Upon launch, Protheus will search for a config.lua file in the same directory as the executable. The configuration will contain the information about the relative location of the game files.

# Graphics

The graphics system of Protheus currently uses SDL to implement OpenGL, Dependencies on SDL are expected to be removed as SDL is just used to speed up the prototype stage.

## Sprite Batcher

The sprite batcher is designed to improve the rendering by using a design which will allow for higher utilisation of OpenGL (not implemented). Condensing render calls into a single large render call will reduce the CPU overhead of rendering. Using a sprite batcher also allows us to move the rendering of the queue into a separate thread.

void **flush**()

Processes all sprites which have been pushed and renders them in first in first out order.

void **push**(Sprite\* *sprite*, Vector4& *position*, float *scale*, float *rotate*)

Push a sprite into the batch queue to be rendered when flushed. The sprite which it is being pointed at must not be destroyed until after being flushed. The scale and rotate are optional with the default values being a scale of 1 and rotation of 0 degrees.

## Sprite Manager

The sprite manager is responsible for loading, releasing and storing sprites loaded into Protheus. The Sprite Manager will provide all sprites with a game\_id which is registered in the GUIDLookup. By default the name of a sprite will be a null string, unless provided with one when loading a sprite. A name must be provided to retrieve the sprite without the game\_id which it was provided with.

Sprite\* getSprite (const game\_id)

Using the sprites game\_id we can request the sprite manager to look for a corresponding sprite and return a pointer to the manager’s version, if a sprite isn’t found then a nullptr is returned.

AnimatedSprite\* getAnim (const game\_id)

Sprite\* loadSprite (const string& *name*, const string& *imagePath*)

void release (const game\_id)

### Animated Sprite

### Sprite

## Renderer

## Window

## Text Renderer

# Audio

# Scripting