VG Engine

Technical Document

Author: Joel Mitrunen, Miika Silvola, Tuomo Oksanen, Kari Tirkkonen, Toni Sarvela

Date: 4.12.2015

Confidentiality: Private

Version: 1.2.2

**Table of Contents**

1 Version history [3](#__RefHeading___Toc281913994)

2 Abbrevations, accronyms and key terms [3](#__RefHeading___Toc281913995)

3 Introduction [3](#__RefHeading__581_155240451)

3.1 The purpose of this document [3](#__RefHeading___Toc281913997)

3.2 Content of this document [3](#__RefHeading___Toc281913998)

4 SW REQUIREMENTS [4](#__RefHeading___Toc281913999)

4.1 Graphics [4](#__RefHeading___Toc281914000)

4.2 Physics [4](#__RefHeading___Toc2819140001)

4.3 Audio [4](#__RefHeading___Toc2819140002)

4.4 Input [4](#__RefHeading___Toc28191400022)

4.5 Frameworks / portability [4](#__RefHeading___Toc28191400023)

4.6 Scene management [4](#__RefHeading___Toc281914000231)

5 Architecture [5](#__RefHeading__583_155240451)

6 Subsystems [6](#__RefHeading__585_155240451)

6.1 Graphics [6](#__RefHeading__587_155240451)

6.2 Physics [6](#__RefHeading___Toc28191400011)

6.3 Audio [6](#__RefHeading___Toc28191400021)

6.4 Input [6](#__RefHeading___Toc281914000221)

7 SW Behaviour [7](#__RefHeading__589_155240451)

7.1 App life cycle [7](#__RefHeading__591_155240451)

7.2 Creating of game objects [7](#__RefHeading___Toc281914000111)

# Version history

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Revision** | **Author** | **Comment** |
| 06.02.2015 | 0.0.1 | VG Engine team | Project started |
| 04.05.2015 | 1.0.0 | VG Engine team | First stable version of the engine released |
| 20.2.2015 | 1.1.0 | VG Engine team | Various rendering fixes, scene overhaul, physics and animation added. Windows port added for development purposes. |
| 30.10.2015 | 1.2.0 | VG Engine team | Fixes on physics, rendering and general stability. Camera support added and sounds for windows port. |
| 11.10.2015 | 1.2.1 | VG Engine team | Minor fixes on physics, utility and general stability |

# Abbrevations, accronyms and key terms

|  |  |
| --- | --- |
| **Term** | **Description** |
| SW | Software |
| OpenGL ES | OpenGL for Embedded Systems |
|  |  |

# Introduction

## The purpose of this document

The purpose of this document is to clarify the SW Architecture and describe the engine functionalities. The engine as broken down to subsystem categories, to clarify each subsystem's architecture and their capabilities. This document does not provide any code examples how the mechanics are implemented but to give a detailed overview of on general level. The document also describes the class diagram of each subsystem.

# VG Engine Features

The game engine provides following features for game developer.... It is enough to have only short list of different features, which SW provides. One or two pages, which describes the main features briefly.

## Graphics

VG Engine offers all basic graphical properties that are need in developing a game, such as: textures, animations, color adjustment, rotations, texts and resizing. OpenGL is used to draw the game objects on windows (Glew) and OpenGL ES on android devices.

## Physics

Features offer creating polygon, circle and custom physics shapes. You can combine 2 different physics components together with physics joints. Material properties such as setting a friction, restitution and density of an object. Collision detection and collision filtering are also supported. You can also change physics world gravity and physics update interval. Ability to change objects velocity and give them force.

## Audio

Audio has a lot of same code in both Windows and Android. Audio is done with OpenSL ES on Android and irrKlang on Windows. Both platforms support playing, pausing, stopping, seeking and looping audio. You can also play multiple sounds at once on both platforms.

## Input

The input features are implemented for both android touch screen devices and Windows keyboard/mouse. For android system, touch input offers only single touch functions only. There is also sensor input support for android which includes accelerometer and gyroscope data handling. Windows version supports all keyboard buttons and their states (up, down, hold, not pressed) and the basic 3 buttons for mouse (Right, left and middle) and their states (up, down, hold, not pressed).

## Frameworks / portability

A port for Windows has already been implemented on the engine, so implementations on other systems should be possible as well.

## Scene management

VG Engine has a game object system, which handles the updating automatically. Developers can give a list of components for the game object, which are each updated by its own corresponding system. There is only one system for each component type and the components are identified in real time.

Game object system goes hand to had with VG Engines scene management system. Scene is a template class for the Game Developers to handle their own scenes in their game. Scene comes with a list of game objects, which are added to the scene with adding functions. Scene updating and game mechanics are handled by scene system. Scene system is to be implemented by game developers.

# Architecture

Engine architecture is featured below in the next titles.

# Subsystems

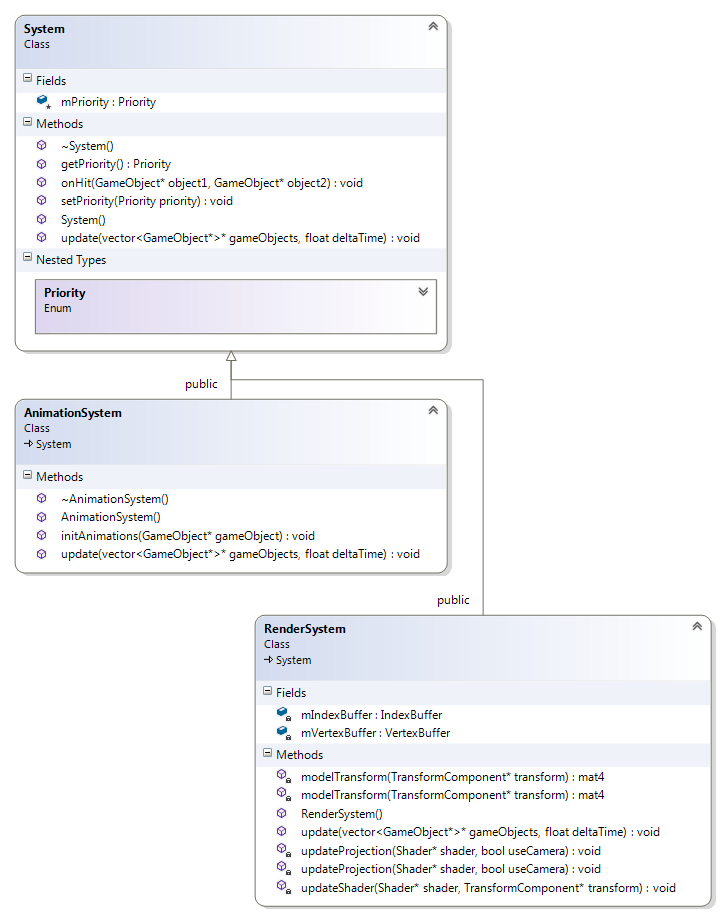
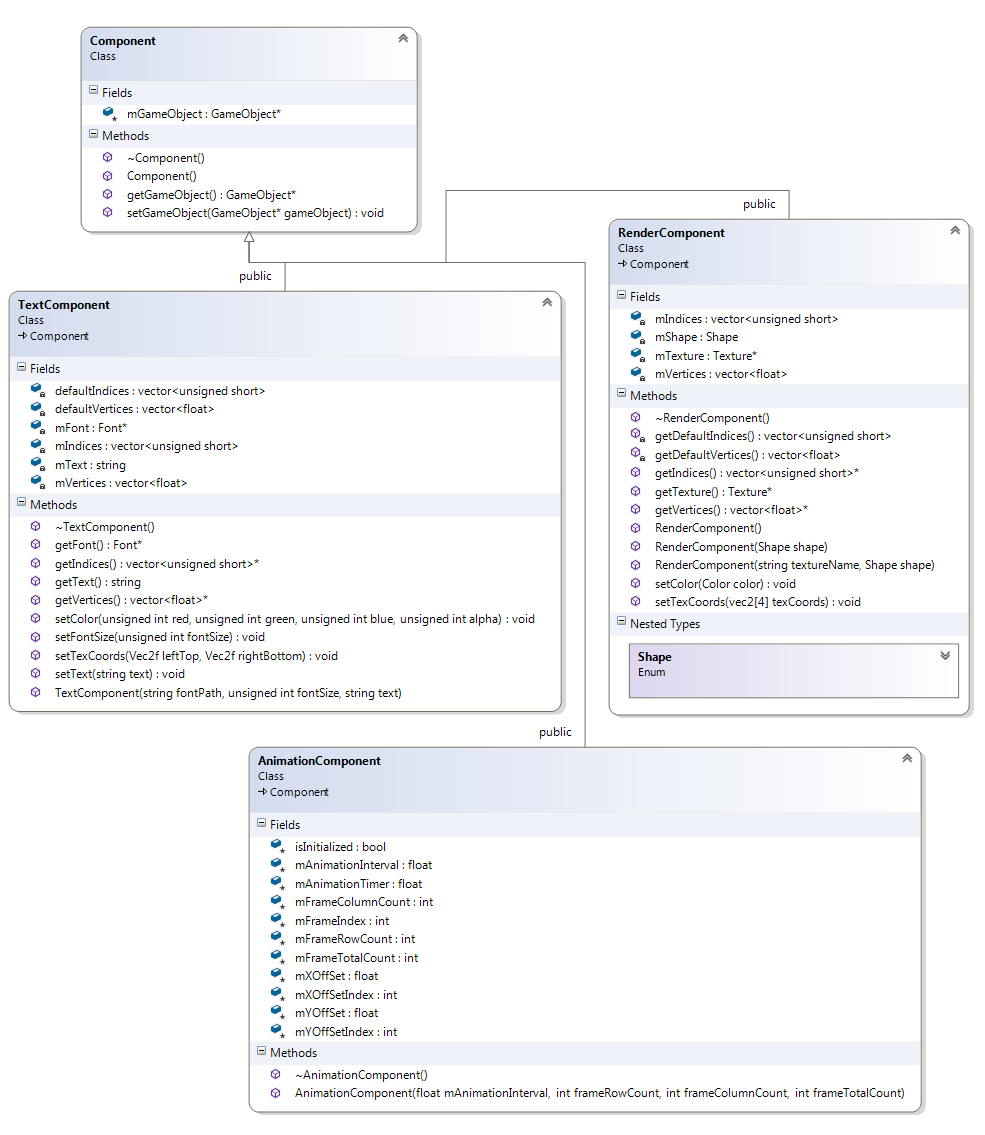
In this section, describe actual class diagrams etc. for each subsystem in separate sections like below:

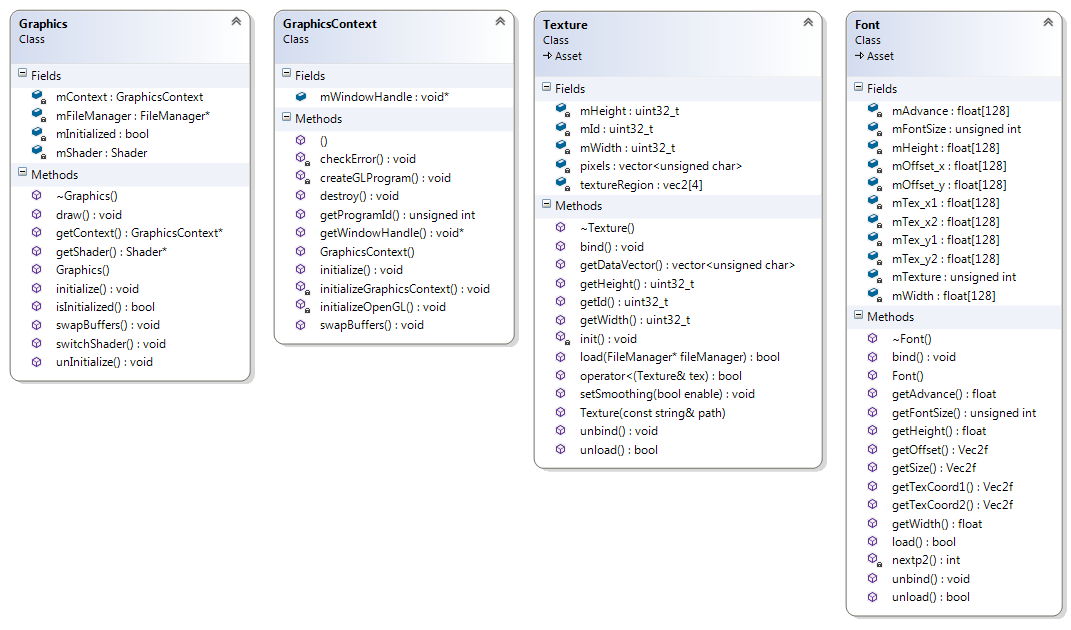
## Graphics

Graphics class contains and manages GraphicsContext and Shader. GraphicsContext is initialized and Shader is loaded at launch. Screen size is saved in the static Screen class.

RenderSystem is added to all new Scenes by default. It will draw all GameObjests with at least a TransformComponent and a RenderComponent or a TextComponent. If the RenderComponent has a Texture it will be bind before drawing. The same is done for the Font that the TextComponent uses. AnimationSystem is also added to all new Scenes. It updates all GameObjects with at least a RenderComponent and an AnimationComponent.

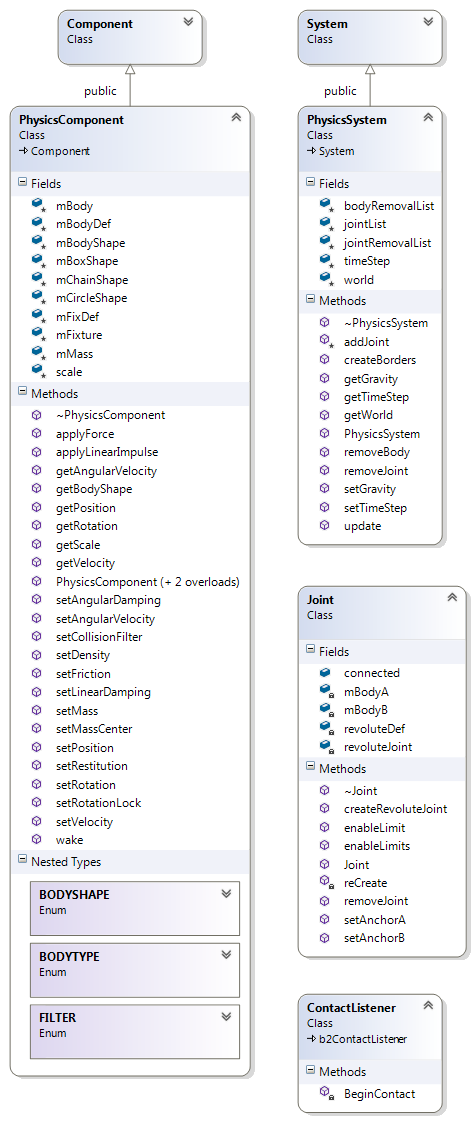
The static Camera class contains values that are added to transform matrices before rendering. Each TransformComponent defines if their position and size are relative to Camera or not.





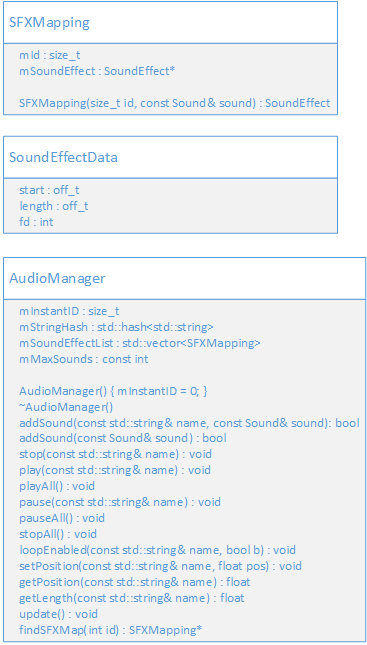
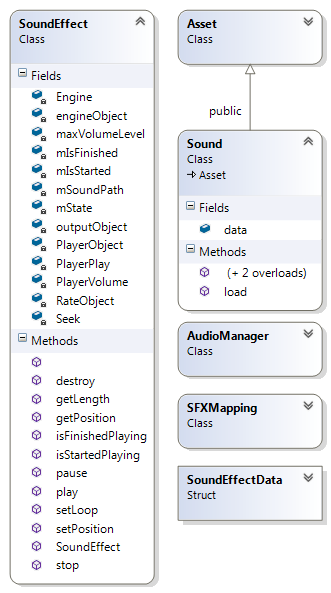
## Physics

VG Engine uses Box2D Physics Engine. GameObjects that have at least Transform and Physics Components get updated by PhysicsSystem. Values in TransformComponent get Replaced with values from Box2D world.



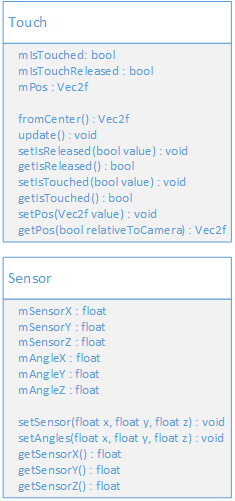
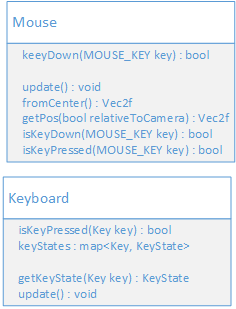
## Audio

VGEngine uses OpenSL ES audiolibrary to produce sound. Allows features such as pausing, continuing and loop of a sound file.



## Input

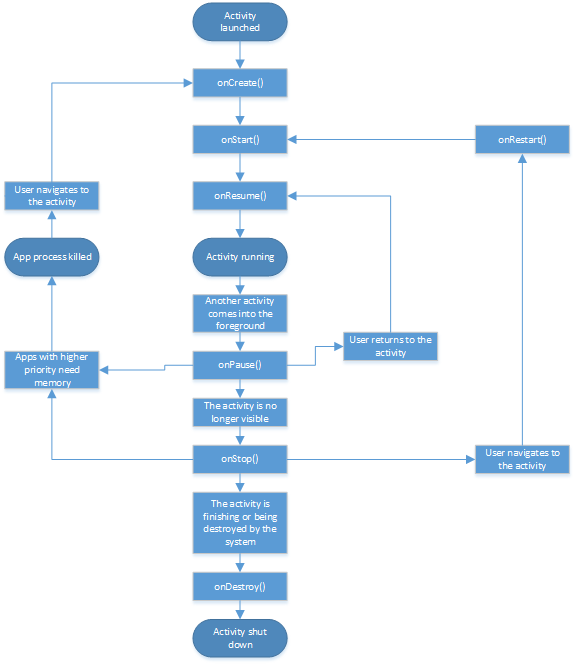
Input states and the last state are saved at beginning of the update loop. This allows detecting a single input when a key or a finger on touch screen is held down.



# SW Behaviour

## App life cycle

Android NDK life cycle



The Native activity architecture is established on events.

* onCreate() starts the activity and initializes all objects.
* onStart() is a initialization finished flag.
* onPause() is a command which is give to the activity when another activity has been started
* onResume() resumes running the activity which appears on main screen of the android device
* onStop() activity flagged for destruction.

## Creating of game objects

