Mikko Romppainen

xplatform\_framework

**Ohjelmistosuunnitelma**

12.10.2017

Laatija/t: Mikko Romppainen

Luottamuksellisuus: julkinen

Hyväksynyt: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Versio: 1.0.0

Sisällys

[1 Muutoshistoria 1](#_Toc495570858)

[2 Johdanto 2](#_Toc495570859)

[2.1 Dokumentin tarkoitus 2](#_Toc495570860)

[2.2 Dokumentin sisältö 2](#_Toc495570861)

[3 Määritelmät ja termien selitykset 3](#_Toc495570862)

[4 Ohjelmistoarkkitehtuuri 4](#_Toc495570863)

[4.1 Object 5](#_Toc495570864)

[4.2 GraphicsApplication 5](#_Toc495570865)

[4.3 Window 6](#_Toc495570866)

[4.3.1 Win32Window 7](#_Toc495570867)

[4.3.2 AndroidWindow 7](#_Toc495570868)

[4.4 GraphicsSystem 8](#_Toc495570869)

[4.4.1 OGLGraphicsSystem 9](#_Toc495570870)

[5 Ajoympäristö 10](#_Toc495570871)

[5.1 Hakemistorakenne 10](#_Toc495570872)

[5.2 Frameworkin kääntäminen ja esimerkkisovelluksen testaaminen 10](#_Toc495570873)

[5.2.1 Windows 10](#_Toc495570874)

[5.2.2 Android 10](#_Toc495570875)

# Muutoshistoria

Dokumentin muutoshistoria on esitetty taulukossa 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Henkilö | Päiväys | Versio | Kommentti |
| Mikko Romppainen | 12.10.2017 | 1.0.0 | Dokumentti luotu. Ensimmäinen virallinen release-versio. |

Taulukko 1. Muutoshistoriataulukko

# Johdanto

## Dokumentin tarkoitus

Dokumentti on tekninen kuvaus xplatform\_frameworkin toiminnasta. xplatform\_framework on valmis sovelluskehys, jonka avulla on OpenGL ES 2.0 pohjaisten sovellusten tekeminen helpompaa. Sovelluskehys tarjoaa muutaman perusluokan, joiden vastuulla on mm. sovelluksen pääikkunan luominen ja OpenGL ES 2.0 laitteen alustustoimet. Lisäksi sovelluskehys tarjoaa kantaluokan käyttäjän omia sovelluksia varten.

## Dokumentin sisältö

Dokumentin kappaleessa 4 on kerrottuna sovelluskehyksen arkkitehtuuri ja kaikki sovelluskehyksen tarjoamat luokat ja niiden merkityksen. Kappaleessa 5 on esitettynä projektin kansiorakenne ja kansioiden käyttötarkoitus. Lisäksi Kappaleessa 5 on ohjeet, kuinka sovelluskehyksen voi kääntää eri tuetuille alustoille.

# Määritelmät ja termien selitykset

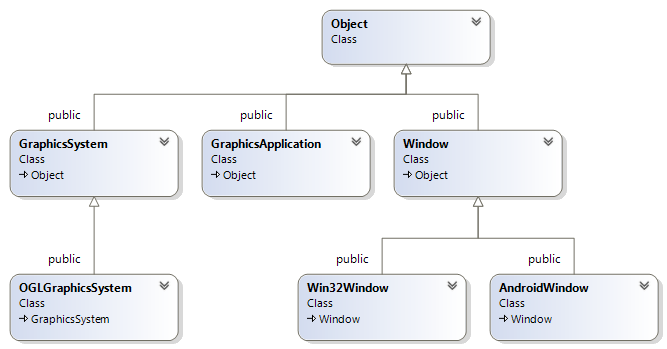
Dokumentissa käytetyt termit ja lyhenteet on selitetty taulukossa 2.

|  |  |
| --- | --- |
| Termi | Kuvaus |
| Framework | Sovelluskehys. Valmis runko ohjelmistolle, jonka on tarkoitus helpottaa ja nopeuttaa sovelluksen tekemistä. |

Taulukko 2. Selitystaulukko

# Ohjelmistoarkkitehtuuri

Sovelluskehyksen ohjelmistoarkkitehtuuri on esitetty kuvassa 1.



Kuva 1. Sovelluskehyksen arkkitehtuuri

Sovelluskehys koostuu muutamasta tärkeimmästä luokasta. Object-luokka, jota käytetään jokaisen frameworkin luokan kantaluokkana. Sovelluskehys tarjoaa käyttäjälle valmiin ikkunan sovellukselle, ja esimerkin omaisen luokan Grafiikkalaitteiston abstrahoimiseksi. Lisäksi sovelluskehyksessä on luokka, jota käyttäjän pitää käyttää omien sovellusten kantaluokkana. Jokainen luokka on dokumentoitu tarkemmin dokumentin seuraavissa kappaleissa.

## Object

Object-luokan luokkakaavio on esitetty kuvassa 2. Object luokka toimii kantaluokkana jokaiselle frameworkin luokalle. Object-luokka tarjoaa metodit objektin viittauslaskennan toteuttamiseksi. Tulevaisuudessa Object-luokkaa voidaan myös käyttää oman muistiallokaattoreiden toteuttamiseksi, joten Object-luokan käyttäminen frameworkilla tehdyissä sovelluksissa on suotavaa.



Kuva 2. Object-luokan luokkakaavio

Object-luokka toteuttaa sisäisesti myös hieman yksinkertaista muistinhallintaa, jolloin vapauttamattomat Object-instanssit tunnistetaan ohjelmiston lopussa ja käyttäjälle annetaan virheilmoitus, mikäli joitakin Object-luokan tai siitä johdettujen luokkien instansseja on vapauttamatta ohjelman suorituksen lopussa.

Objektien yhteydessä voidaan myös käyttää Ref-luokkaa, joka tarjoaa vastineen ns autopointtereille. Auto-pointterien/Ref-luokan käyttäminen on suotavaa, koska silloin tyypillisesti koodataan vähemmän muistivuotoja.

## GraphicsApplication

GraphicsApplication-luokan luokkakaavio on esitetty kuvassa 3. GraphicsApplication-luokan päätarkoitus on abstrahoida Sovellus-luokka ja tarjota virtuaaliset update ja render –metodit, joiden avulla käyttäjä suorittaa sovelluksen aikastepissä tarvittavan päivittämisen (update-metodi) ja ruudun piirtämisen (render-metodi). GraphicsApplication pitää sisällään myös objektien instanssit Window ja GraphicsSystem-tyyppisiin olioihin, joiden avulla käyttäjän toteuttama sovellus voi piirtää ruudulle ja kysellä tietoja mm. ruudun koosta.



Kuva 3. GraphicsApplication-luokan luokkakaavio.

Tehdäkseen oman sovelluksen, frameworkin käyttäjän täytyy tehdä oma luokka, jonka kantaluokaksi pitää määrittää GraphicsApplication. Johdetussa sovellus-luokassa täytyy olla toteutus kantaluokan abstrakteille metodeille render ja update. GraphicsApplication-luokan käyttöön löytyy esimerkki kansiosta ”GraphicsTest”, luokka ”TestApplication”.

## Window



Kuva . Window-luokan luokkakaavio

### Win32Window



Kuva . Win32Window-luokan luokkakaavio.

### AndroidWindow



Kuva . AndroidWindow-luokan luokkakaavio.

## GraphicsSystem



Kuva . GraphicsSystem-luokan luokkakaavio.

### OGLGraphicsSystem

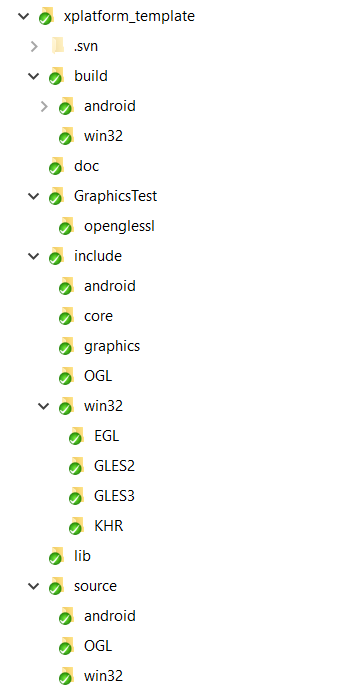


Kuva . OGLGraphicsSystem-luokan luokkakaavio.

# Ajoympäristö

## Hakemistorakenne

Projektin kansiorakenne on esitetty kuvassa 9.



Kuva . Projektin kansiorakenne.

Tässä listaa muutamista tärkeimmistä kansioista:

**build\android** – Android Studio -yhteensopivat projektitiedostot Android käännöksen tekemiseen.

**build\win32** – Visual Studio 2015 -yhteensopivat projektitiedostot Windows käännöksen tekemiseen.

**include** – Sisältää header-tiedostot frameworkin luokille. Tiedostot on jaoteltu eri alikansioihin include-kansion sisällä.

**lib** – Sisältää staattisia ulkoisia kirjastoja, joita tarvitaan frameworkin buildaamiseen erityisesti win32 ympäristöön.

**source** – Sisältää lähdekoodit frameworkin luokille. Tiedostot on jaoteltu eri alikansioihin source-kansion sisällä

**doc** – Tämä kansio sisältää frameworkin dokumentaation.

## Frameworkin kääntäminen ja esimerkkisovelluksen testaaminen

Tässä kappaleessa on kerrottu lyhyesti, kuinka sovelluskehyksen kääntäminen ja esimerkkisovelluksen suorittaminen onnistuu eri alustoilla.

### Windows

Esiehdot: Microsoft Visual Studio 2015 on asennettuna koneelle.

Windows käännöksen tekeminen onnistuu helpoiten avaamalla **build\win32\engine.sln** -tiedosto Visual Studio 2015:llä. Kun projekti on avattu, itse kääntäminen onnistuu tuttuun tapaan valitsemalla valikosta ”**Build -> Rebuild Solution**”. Esimerkkisovelluksen käynnistäminen onnistuu tämän jälkeen valitsemalla ”**Debug -> Start without debugging**”.

### Android

Esiehdot: Android Studio on asennettuna koneelle.

Android käännöksen tekeminen onnistuu helpoiten avaamalla kansio: ”**build\android**” Android Studioon. Kun projekti on avattu, onnistuu käännöksen tekeminen valitsemalla ”TODO”. Sovelluksen käynnistäminen joko puhelimeen tai emulaattoriin onnistuu valitsemalla valikosta ”TODO”.