1 LIGBDX Explicacion Framework

1.1 Sumario

1.2 Sitios de obrigada visita

Neste punto vos amoso aqueles sitios de Internet que seguramente necesitaredes visitar.

- API do LIBGDX: Todas as clases coas definicións dos seus métodos que utiliza o framework LIBGDX. http://libgdx.badlogicgames.com/nightlies/docs/api/
- WIKI do LIBGDX: https://github.com/libgdx/libgdx/wiki
- Foro do LIBGDX: http://www.badlogicgames.com/forum

1.3 Framework vs Motor de xogos

Aínda que o curso é eminentemente práctico neste punto imos explicar en que consiste o framework e que é o que vai xerar.

Primeiro deberemos explicar que é un framework. Ben, case vou dicir que un framework non é un motor de xogos e traducido a nosa linguaxe significa que imos ter máis traballo á hora de programar un videoxogo, pero tamén imos ter moita máis liberdade no seu desenvolvemento.

Un framework é un conxunto de librerías que nos permiten (neste caso) o desenvolvemento de xogos facilitándonos a labor de programación sobre os diferentes aspectos no deseño dos mesmos como poden ser: gráficos, son é música, renderizado, xestión de eventos,...Por exemplo (a través de programación): usa a clase Texture para cargar un gráfico; sitúa este gráfico nesta posición e move dende esta posición a esta posición de forma continua (todo por programación).

Un motor de xogos actúa nun nivel superior. É un software completo no que nos 'dámoslle' os datos para unha determinada funcionalidade a un nivel moi abstracto e el se encarga de xerar todo o necesario para que funcione. Normalmente son máis 'intuitivos' (están cheos de ferramentas gráficas). Por exemplo, poderíamos cargar un gráfico e decirlle: quero que movas este gráfico desta posición a esta posición de forma continua. A carga do gráfico faríase graficamente(:)) e dita funcionalidade tamén poderíase indicar graficamente.

Por que usar un framework e non un motor de xogos ?

- Tendo en conta que este curso forma parte da programación do módulo 'Programación de dispositivos móbiles' do ciclo Desenvolvemente de aplicacións multiplataforma, necesitaba algo que tivera unha 'continuidade' para os alumnos. En programación usan Java e é a linguaxe usada por este framework. Ó ser un framework obriga a programar, cousa que non motor de xogos non tería tanto peso.

Por que usar LIBGDX e non outro calquera ?

- Primeiro pola linguaxe usada, que é JAVA igual que a que utilizan no módulo de programación.
- Segundo porque é gratuito. Moitos motores e framework teñen unha versión gratuíta pero para ter todas as funcionalidades ou para 'vender' o xogo hai que pagar.
- Terceiro porque serve para crear xogos 2D e 3D.
- Carto é multiplataforma. Probar xogos co emulador de Android é imposible. Se necesitaría unha máquina virtual con Android instalado complicando máis o proceso de probas. Desta forma podemos desenvolver o xogo no PC e cando estea rematado probalo nun dispositivo Android.

Existen outros framework que teñan todo o anterior ?

- Pois sí. Tedes unha comparativa bastante completa na WIKI.

1.4 Framework LIBGDX

É un framework para o desenvolvemento de xogos coa linguaxe Java tendo coma principal vantaxe que é multiplataforma. Para nós iso significa que podemos desenvolver o xogo independentemente da plataforma onde se vaia a executar.

Así grazas a este framework podemos desenvolver xogos para:

- Windows
- Linux
- Mac OS X
- Android (+2.2)
- BlackBerry
- iOS
- Java Applet (require a máquina virual java instalada)
- JavaScript/WebGL (Google Chrome, Safari, Opera, Mozilla Firefox, IE via Google Chrome Frame)

Máis información sobre as vantaxes en: http://libgdx.badlogicgames.com/features.html

Polo tanto imos poder desenvolver un xogo e probalo no PC sen necesidade de utilizar un dispositivo móbil (unicamente que o xogo se basee nalgún hardware específico deste tipo de dispositivos coma poda ser o acelerómetro)

1.5 Compoñentes

LibGDX está composto por unha serie de compoñentes:

- Marco de Aplicación, que manexará o bucle principal y ademais estará encargado do ciclo de vida, é dicir, os eventos de creación, destrución, pausa e resume.
- Un compoñente de Gráficos que permitiranos xestionar a representación de imaxes e obxectos gráficos na pantalla.
- Un compoñente de Audio, que facilitaranos o acceso ós sons e música da aplicación.
- Un compoñente de Entrada e Saída para ler e escribir os diferentes arquivos de datos como por exemplo, arquivos de configuración, sons, música, texturas,...
- Un compoñente de Entrada que xestionará a entrada a través do teclado, pantalla táctil ou acelerómetro.

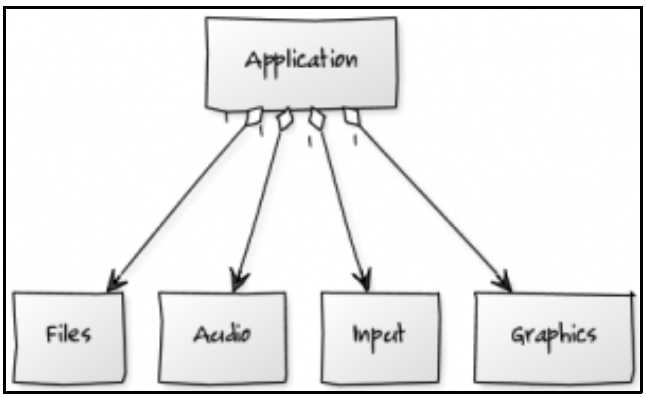


Gráfico obtido deste documento

Tamén temos módulos específicos para a xestión de cálculos matemáticos, coma Math e tamén módulos para a xestión de colisións entre os elementos gráficos dos xogos.

Outro gráfico que amosa de diferente forma a estrutura de módulos do framework LIBGDX:

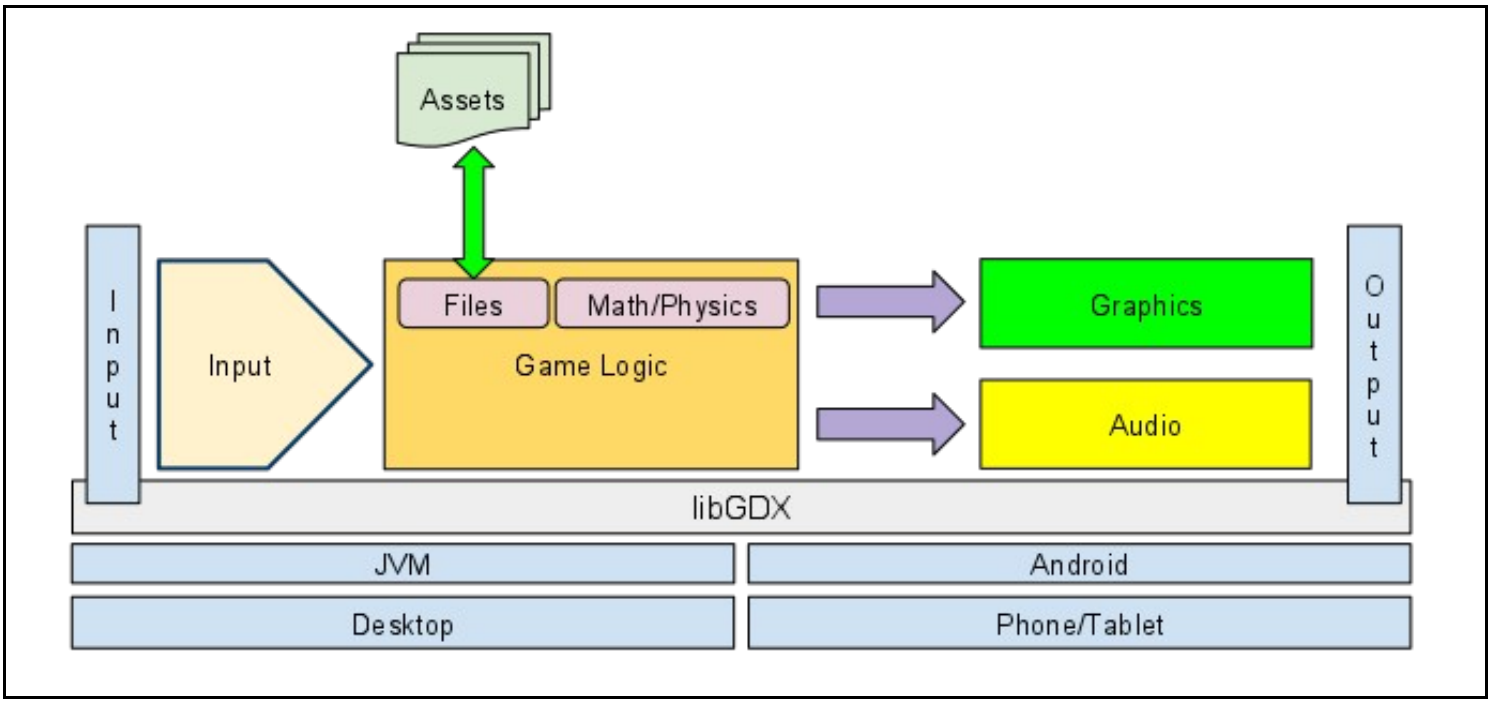


Gráfico obtido deste documento

Nota: Assets é o cartafol onde imos gardar todos os recursos (gráficos ? modelos 3d? sons ? música,...) do noso xogo. Nas aplicacións Android normalmente se gardaría no cartafol /res. Podemos ver neste enlace a diferenza entre os dous cartafoles.

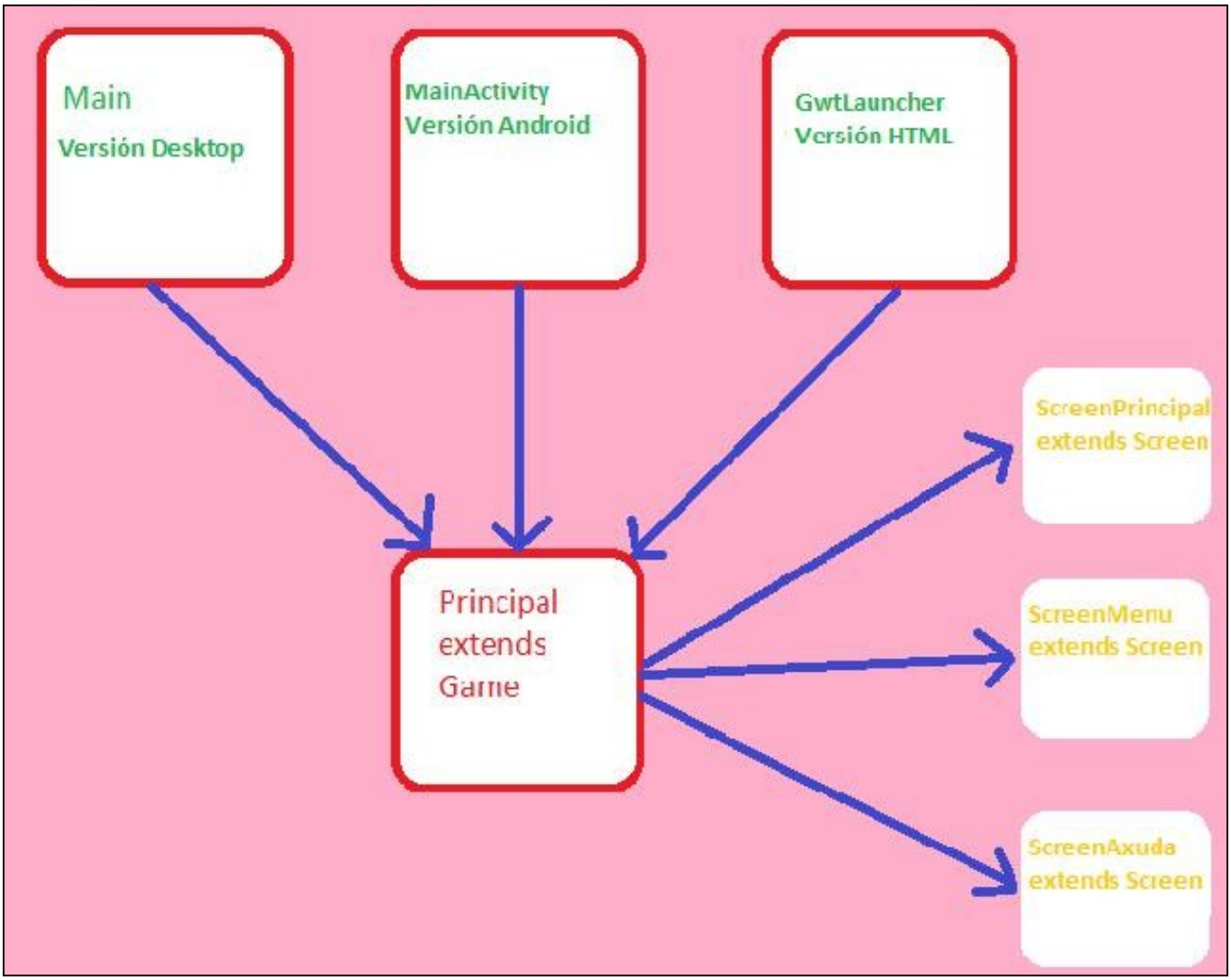
En Android, os gráficos gardados en /res se 'compilan' co proxecto e podemos facer referencia a eles a través de identificadores. No cartafol /Assets podemos crear unha estrutura de cartafoles, cousa que non podemos facer en /res.

1.6 Proxectos que vai xerar o framework LIBGDX

É moi importante que comprendamos que usando unha ferramenta gráfica, imos xerar un proxecto de eclipse por cada plataforma de xogo e un proxecto común que farán uso del todos os demais proxectos.

Os proxectos xerados de cada plataforma o único que farán será chamar a unha clase (que vai derivar da clase Game) para comezar o xogo. Así, se vou desenvolver un xogo para PC (Windows / Linux) e Android, a ferramenta vaime xerar 3 proxectos: un para Android, outro para PC e outro 'Comun' onde estará toda a programación do videoxogo. Os proxectos Android e PC só chamarán a unha clase do proxecto común e a partires de entón o control estará todo nese proxecto.

É dentro deste proxecto común onde implantaremos toda a lóxica do xogo, renderizado, uso de música e son, xestión de eventos. Teremos varias pantallas (polo xeral), como unha pantalla de presentación, do xogo, high scores,...(derivarán da clase Screen).



-- Ángel D. Fernández González -- (2014).