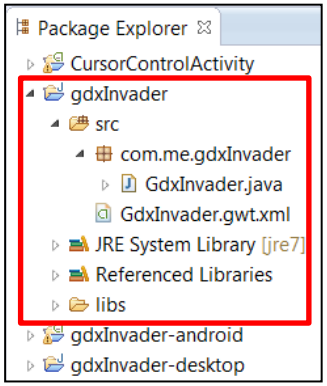
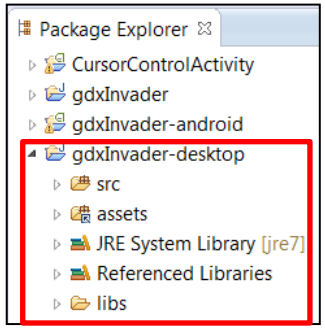
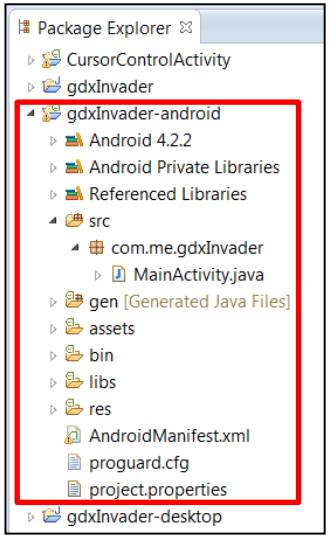
La structure

Libgdx : la structure

- Plusieurs parties
- Partie Logique
  - Code source du jeu
- Partie Desktop
  - Lanceur de la partie logique pour Desktop.
  - JOGL ou LWJGL (lien OpenGL).
- Partie Android
  - Lanceur de la partie logique pour Android.

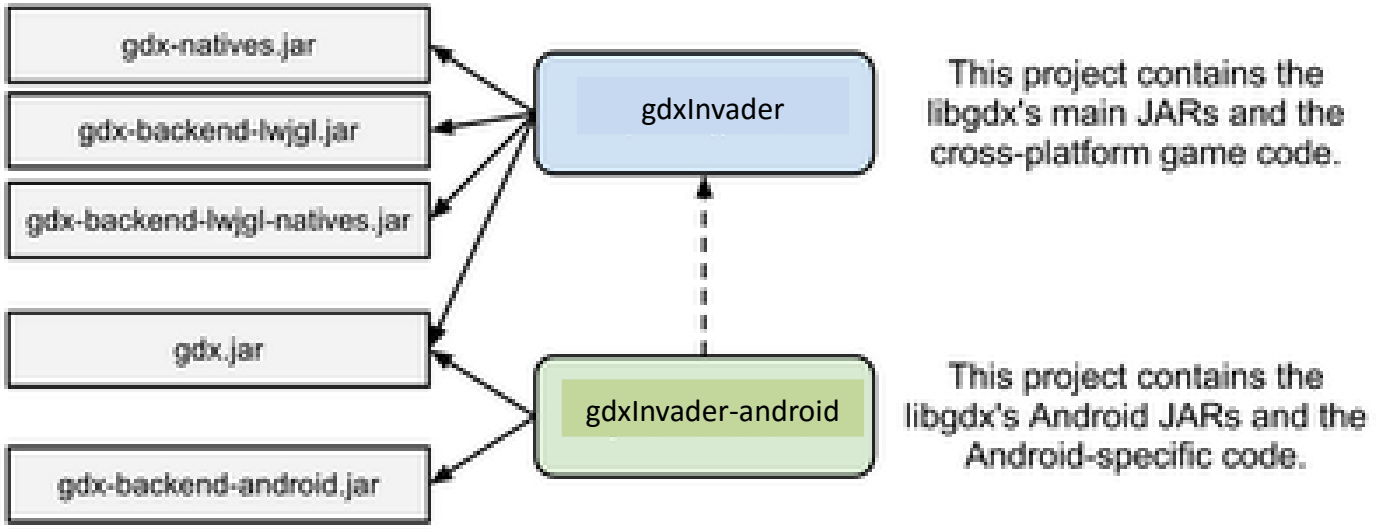






http://www.siteduzero.com/informatique/tutoriels/creez-des-applications-pour-android

Libgdx : la structure



http://www.siteduzero.com/informatique/tutoriels/creez-des-applications-pour-android

Libgdx : les modules

- Application
  - Pour exécuter l'application
  - Pour s'informer sur les événements produits au niveau des applications
    - Par exemple le redimensionnement de la fenêtre.
  - C'est le module qui permet d'effectuer la journalisation (le Logging).
- File
  - Pour lire, écrire, même copier, déplacer et supprimer des fichiers.
- Input
  - Pour détecter les entrées de l'utilisateur
    - Par exemple les événements souris (sous desktop), les événements tactile (sous Android) et même l'accéléromètre.

http://www.siteduzero.com/informatique/tutoriels/creez-des-applications-pour-android

Libgdx : les modules

- Audio
  - Pour lire des effets sonores et de la musique en streaming
  - Pour créer des effets sonores.
- Graphiques
  - Pour manipuler le graphisme et paramétrer le mode
- Usage en Java
  - Champs statiques de la classe Gdx
  - Exemples :
    - Gdx.audio pour le module Audio
    - Gdx.file pour le module File
    - Gdx.app pour le module d'application (!)
  - Exemple de code source :
    - `AudioDevice audioDevice =  GDx.audio.newAudioDevice (44100,faux);`

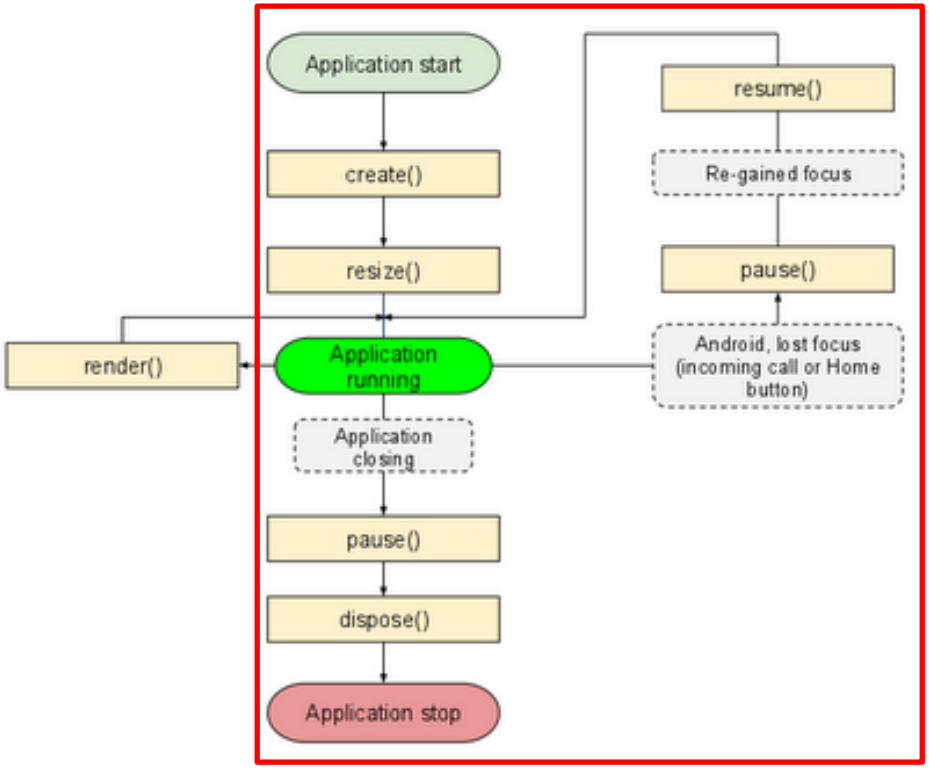
http://www.siteduzero.com/informatique/tutoriels/creez-des-applications-pour-android

Libgdx : le cycle de vie

- Application = suite d'états
  - Créée (ouverte)
  - En cours
  - Suspendue
  - Reprise
  - Détruite (fermée)
- Traitement
  - Interface ApplicationListener
  - Classe ApplicationAdapter

http://www.siteduzero.com/informatique/tutoriels/creez-des-applications-pour-android

Libgdx : le cycle de vie




```
package …;
import com.badlogic.gdx.ApplicationListener;
public class Jeu
      implements ApplicationListener {
  @Override
  public void create() {
  }

  @Override
  public void dispose() {
  }

  @Override
  public void pause() {
  }

  @Override
  public void resize(int arg0, int arg1) {
  }

  @Override
  public void resume() {
  }

  @Override
  public void render() {
  }
}
```

http://www.siteduzero.com/informatique/tutoriels/creez-des-applications-pour-android

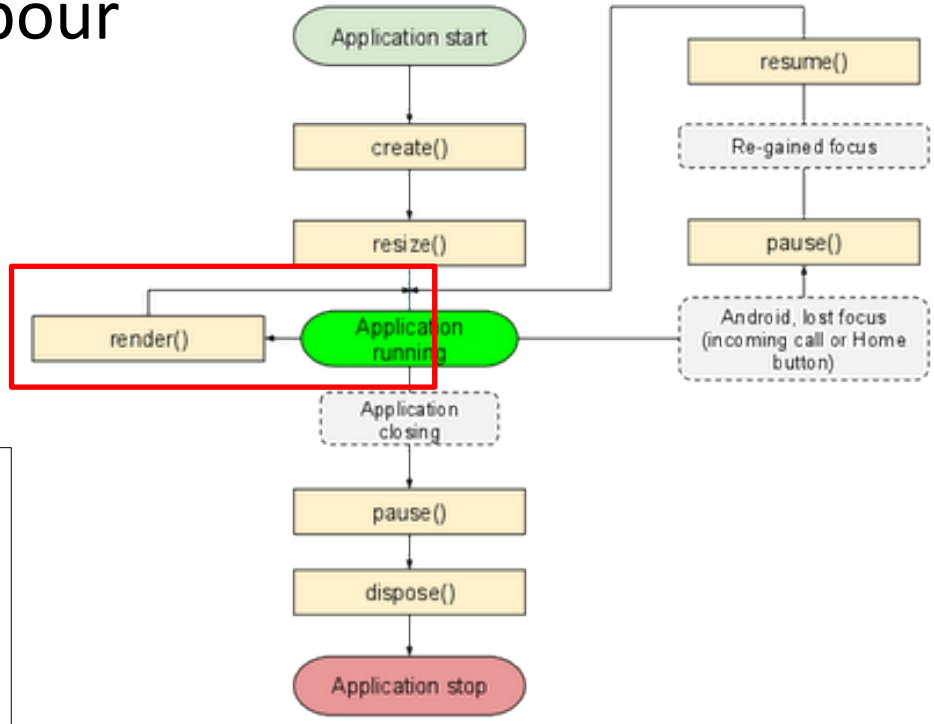
Libgdx : le cycle de vie

- create ()
  - Appelée une fois l'application est créée.
- resize(int width, int height)
  - Appelée à chaque fois que l'écran du jeu (qui n'est pas en état de pause) est redimensionné.
  - Elle est toujours appelée après la méthode create().
  - Les paramètres sont les nouvelles largeur et hauteur(en pixels).
- pause ()
  - Appelée avant dispose ()
  - Android : appelée lorsque le bouton Home est pressé ou un appel entrant est reçu.
  - pause() est un bon endroit pour enregistrer l'état du jeu.
- resume ()
  - Android : appelée lorsque l'application reprend d'un état de pause.
- dispose ()
  - Appelée lorsque l'application est détruite.

http://www.siteduzero.com/informatique/tutoriels/creez-des-applications-pour-android

Libgdx : le cycle de vie

- render ()
  - Appelée en boucle pour chaque rendu.
  - La totalité de la logique du jeu est construite ici.

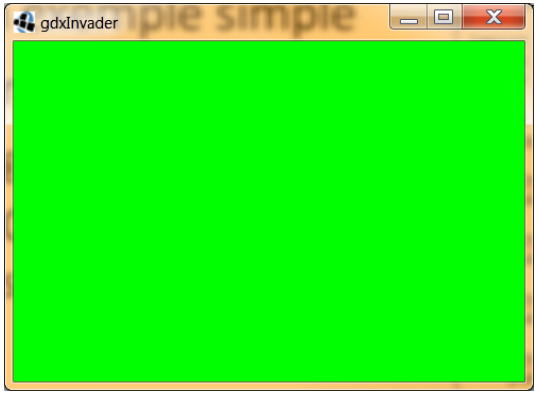


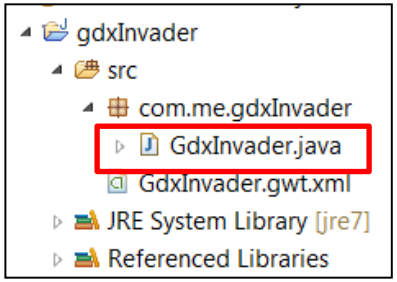

```
package …;
import com.badlogic.gdx.ApplicationListener;
public class Jeu
      implements ApplicationListener {
  …

  @Override
  public void render() {
  }
}
```

http://www.siteduzero.com/informatique/tutoriels/creez-des-applications-pour-android

Libgdx : le cycle de vie

- Un exemple simple
- Partie logique
  - GdxInvader.java :
  - Ce code colorie la surface en vert.



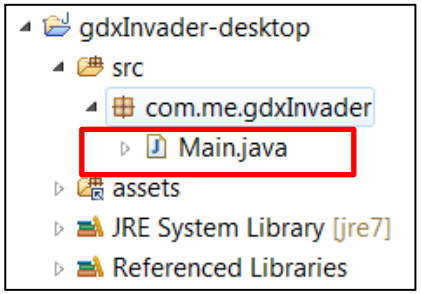



```
package com.me.gdxInvader;

import com.badlogic.gdx.ApplicationListener;
import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.graphics.GL10;

public class GdxInvader
    implements ApplicationListener {

    @Override
    public void create() { }

    @Override
    public void dispose() { }

    @Override
    public void pause() { }

    @Override
    public void resize(int arg0, int arg1) { }


    @Override
    public void resume() { }

    @Override
    public void render() {
       Gdx.gl.glClear(GL10.GL_COLOR_BUFFER_BIT);

       Gdx.gl.glClearColor(0, 1, 0, 1);

    }
}
```

http://www.siteduzero.com/informatique/tutoriels/creez-des-applications-pour-android

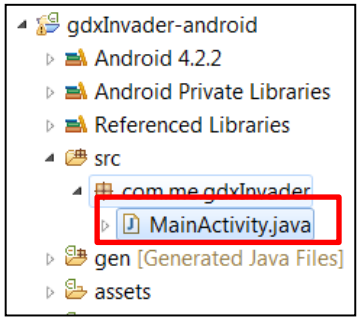
Libgdx : l'exécution d'une application

- Appel à la classe GdxInvader
- Lanceur Desktop




```
public class Main {
  public static void main(String[] args) {
    LwjglApplicationConfiguration cfg = new LwjglApplicationConfiguration();
    cfg.title = "gdxInvader";
    cfg.useGL20 = false;
    cfg.width = 480;
    cfg.height = 320;

    new LwjglApplication(new GdxInvader(), cfg);
  }
}
```

- Lanceur Android




```
public class MainActivity extends AndroidApplication {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    AndroidApplicationConfiguration cfg = new AndroidApplicationConfiguration();
    cfg.useGL20 = false;
    initialize(new GdxInvader(), cfg);
  }
}
```

http://www.siteduzero.com/informatique/tutoriels/creez-des-applications-pour-android

Libgdx : l'organisation

- Séparer le jeu des affichages
- Jeu = Game + { Screens }
- Game
  - Héritage de *com.badlogic.gdx.Game* qui implémente l'interface *ApplicationListener*
- Screen
  - Implémente l'interface *com.badlogic.gdx.Screen*
- Game délègue la gestion des événements au Screen courant
  - setScreen(Screen s);

http://www.siteduzero.com/informatique/tutoriels/creez-des-applications-pour-android

Libgdx : l'organisation

- Notre exemple simple :

```
package com.me.gdxInvader;

import com.badlogic.gdx.Gdx;
import com.badlogic.gdx.Screen;
import com.badlogic.gdx.graphics.GL10;

public class InvaderScreen
    implements Screen {

    @Override
    public void render(float delta) {
       Gdx.gl.glClear(GL10.GL_COLOR_BUFFER_BIT);

       Gdx.gl.glClearColor(0, 1, 0, 1);

    }

    ...
}
```

```
package com.me.gdxInvader;

import com.badlogic.gdx.Game;
import com.badlogic.gdx.graphics.FPSLogger;

public class GdxInvader
    extends Game {

    FPSLogger _fpsLogger;

    @Override
    public void create() {
      _fpsLogger = new FPSLogger ();

      setScreen (new InvaderScreen () );
    }

    @Override
    public void render() {
       super.render ();

       fpsLogger.log ();
    }
}
```

- delta = temps écoulé depuis me dernier render.

http://www.siteduzero.com/informatique/tutoriels/creez-des-applications-pour-android

Libgdx : le « logging »

- Messages pour tracer l'exécution d'un programme
  - Gdx.app.log
- Un message peut être un message normal, d'erreur avec une exception optionnelle ou un message de debug.
  - Gdx.app.log("MyTag", "my informative message");
  - Gdx.app.error("MyTag", "my error message", exception);
  - Gdx.app.debug("MyTag", "my error message");
- En fonction de la plateforme, les messages apparaissent sur la console (desktop) ou le LogCat (Android).
- On peut limiter le logging à un niveau spécifique :
  - Gdx.app.setLogLevel(logLevel);
  - logLevel peut prendre les valeurs
    - Application.LOG_NONE: pas de messages.
    - Application.LOG_DEBUG: tous les messages.
    - Application.LOG_ERROR: seulement les messages d'erreur.
    - Application.LOG_INFO: seulement les messages normaux et d'erreur.

http://www.siteduzero.com/informatique/tutoriels/creez-des-applications-pour-android

Libgdx : le « querying »

- Type d'application
  - Multiplateforme ne veut pas dire code unique

```
switch(Gdx.app.getApplicationType()) {
   case ApplicationType.Android:
       // android specific code
   case ApplicationType.Desktop:
       // desktop specific code
   case ApplicationType.WebGl:
       /// HTML5 specific code
}
```

  - Sur Android, il est également possible de connaître la version du SDK

```
int androidVersion = Gdx.app.getVersion();
```

http://www.siteduzero.com/informatique/tutoriels/creez-des-applications-pour-android

Libgdx : le « querying »

- La mémoire utilisée
  - Java
  - Native

```
int javaHeap = Gdx.app.getJavaHeap();
int nativeHeap = Gdx.app.getNativeHeap();
```

Libgdx : le « threading »

- Les méthodes de ApplicationListener sont appelés par le même thread
- C'est ce thread qui peut appeler OpenGL
- Dans un autre thread = DANGER
  - Ne pas invoquer de commandes OpenGL
  - Créer un objet Runnable :

```
Gdx.app.postRunnable(new Runnable() {
  @Override
  public void run() {
    // code à exécuter dans le thread principal
  }
});
```

http://www.siteduzero.com/informatique/tutoriels/creez-des-applications-pour-android