



26/10/2013

Un moteur de jeu 2D en Java

# Plan

2

- **Moteur** 
- Module Game
- Module Platform
- Module Rts
- Module Network

# Moteur

3

## □ API « bas niveau »

### ▣ Manipulation des ressources

- Visuelles (images, sprites, animations)
- Sonores (sons et musiques)
- Fichiers (binaires et XML)
- Clavier / Souris (curseur Windows / « in-game »)

### ▣ Environnement graphique

- Résolution écran
- Modes de rendu (fenêtré, plein écran, applet)
- Gestion du « frame rate »
- Gestion des séquences (intro, menu, scene...)

# Moteur

4

- API « haut niveau »
  - ▣ Abstraction de premier niveau
    - Classes de base orientées jeux-vidéo généraux
    - Routines de base implémentées et redéfinissables
    - Architecture souple et modulaire
    - Outils standards
  - ▣ Abstraction de deuxième niveau
    - Classes de base dédiées à certains type de jeux-vidéo
      - Jeux de Plateforme
      - Stratégie en temps réel (+pathfinding)

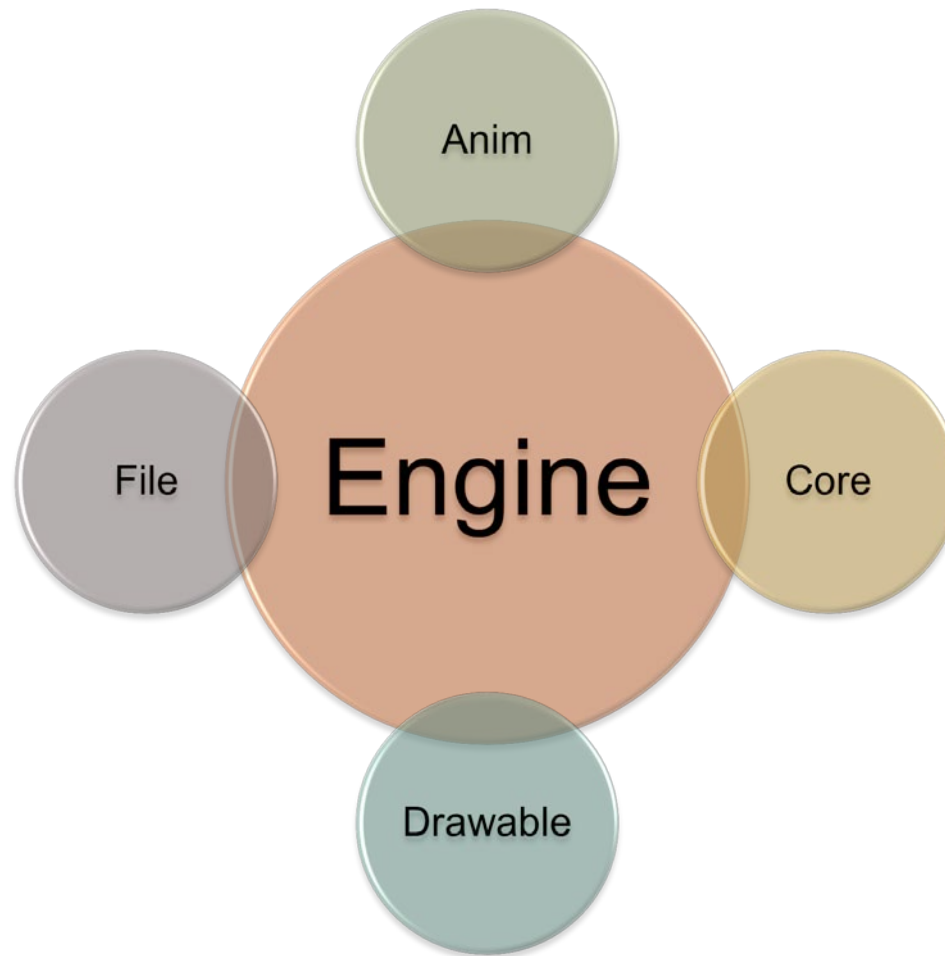
# Moteur

5

- Situé à partir du package: `com.b3dgs.lionengine`
- Principaux packages / classes
  - ▣ `anim` (`Animator`, `Animation`, `AnimState`)
  - ▣ `core` (`Config`, `Engine`, `Graphic`, `Sequence...`)
  - ▣ `drawable` (`Image`, `Sprite`, `SpriteTiled`, `SpriteFont...`)
  - ▣ `file` (`FileReading`, `FileWriting`, `XmlParser`, `XmlNode`)

# Moteur

6



# Moteur - Engine

7

- Squelette de base
  - ▣ `load( ) ;`
  - ▣ `update(double extrp) ;`
  - ▣ `render(Graphic g) ;`
  - ▣ `onTerminate() ; // Optionnel`
- Gestion du nombre d'images par seconde
- Gestion de l'extrapolation ( 'machine independant' )
- Modes d'affichage: `plein écran, fenêtré, applet`

# Moteur - Engine

8



load

- Initialisation des variables
- Chargement des ressources

update

- Mise à jour des variables
- Mise à jour des composants

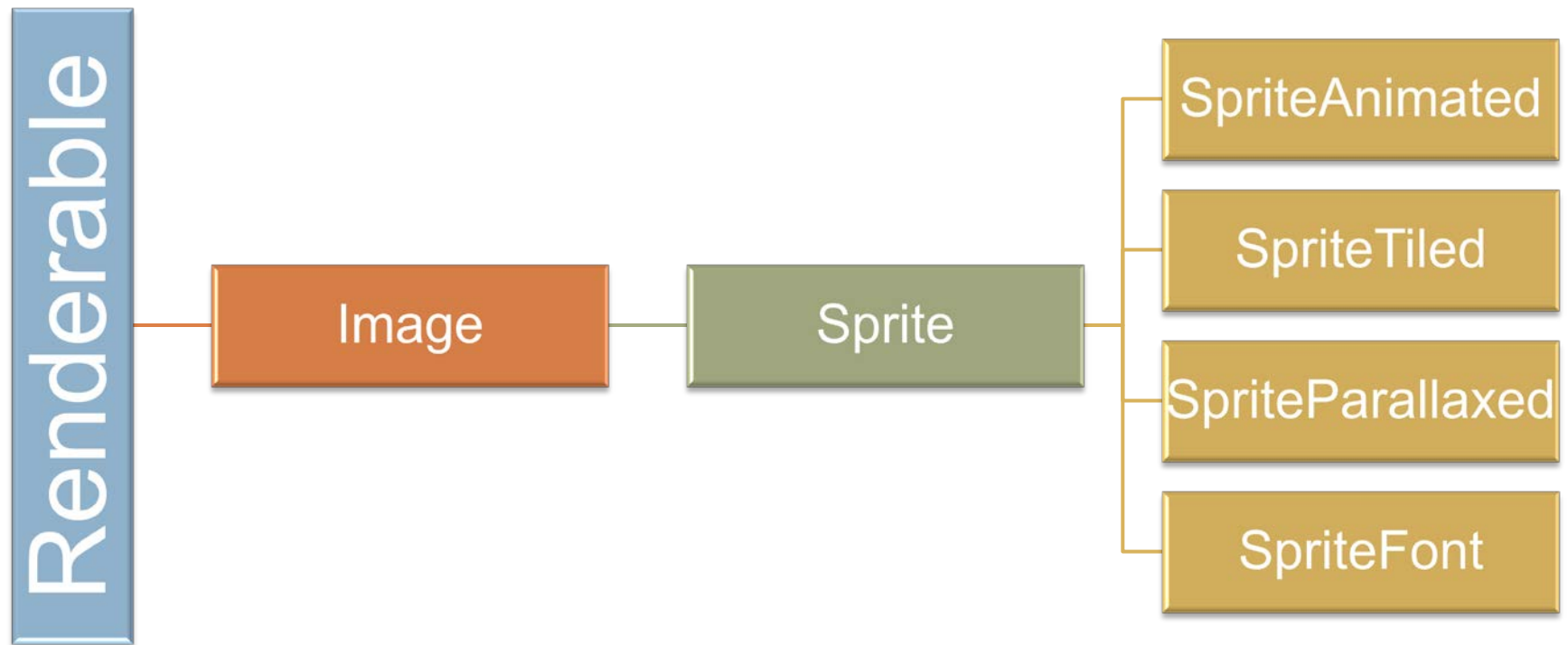
render

- Rendu dans un buffer
- Affichage du buffer à l'écran



# Moteur - Drawable

9



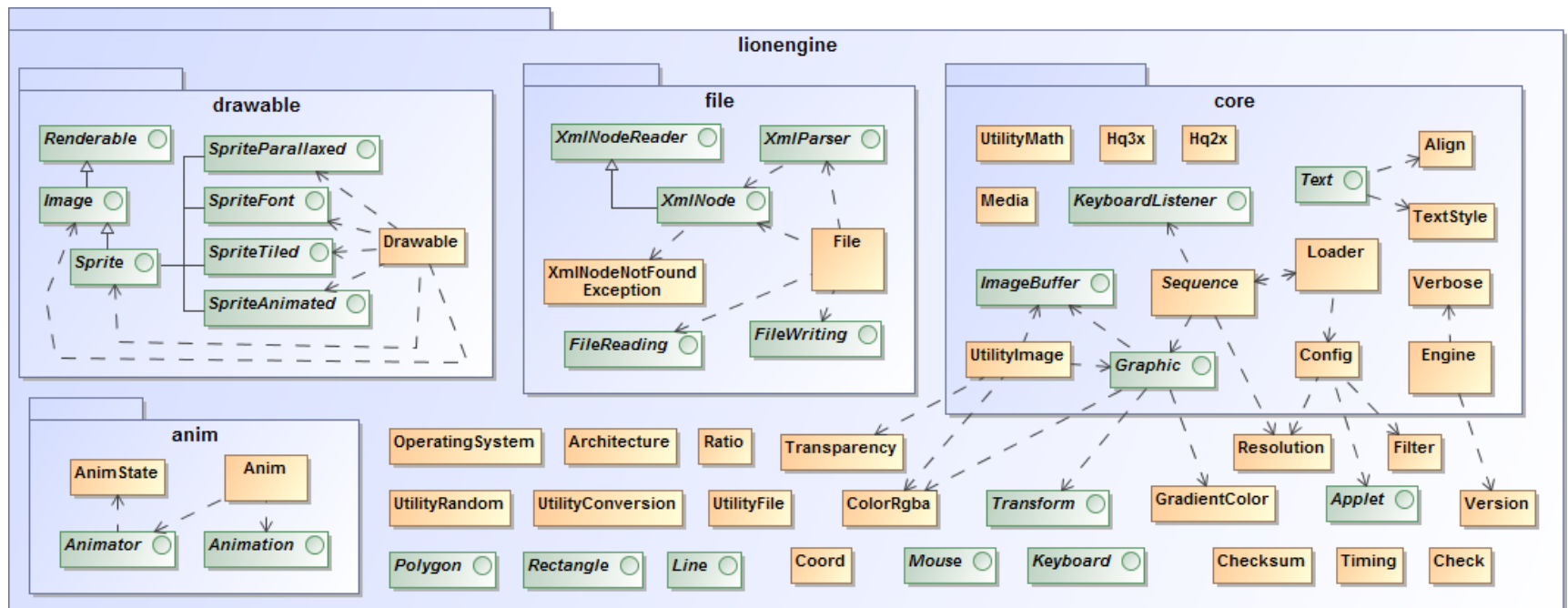
# Moteur - Drawable

10

- **Renderable** (élément affichable simplement)
  - ▣ **Image** (surface non modifiable)
    - **Sprite** (surface modifiable)
      - **SpriteAnimated** (surface animée)
      - **SpriteTiled** (surface découpée en carrés)
      - **SpriteParallaxed** (surface pour un effet 2.5D)
      - **SpriteFont** (police d'écriture depuis une image)

# Moteur - UML

11



# Moteur - Modules

12

- Le moteur complet est composé:
  - ▣ D'une partie centrale
    - Anim
    - Drawable
    - File
    - ...
  - ▣ De modules abstraits
    - Platform
    - Rts
    - ...

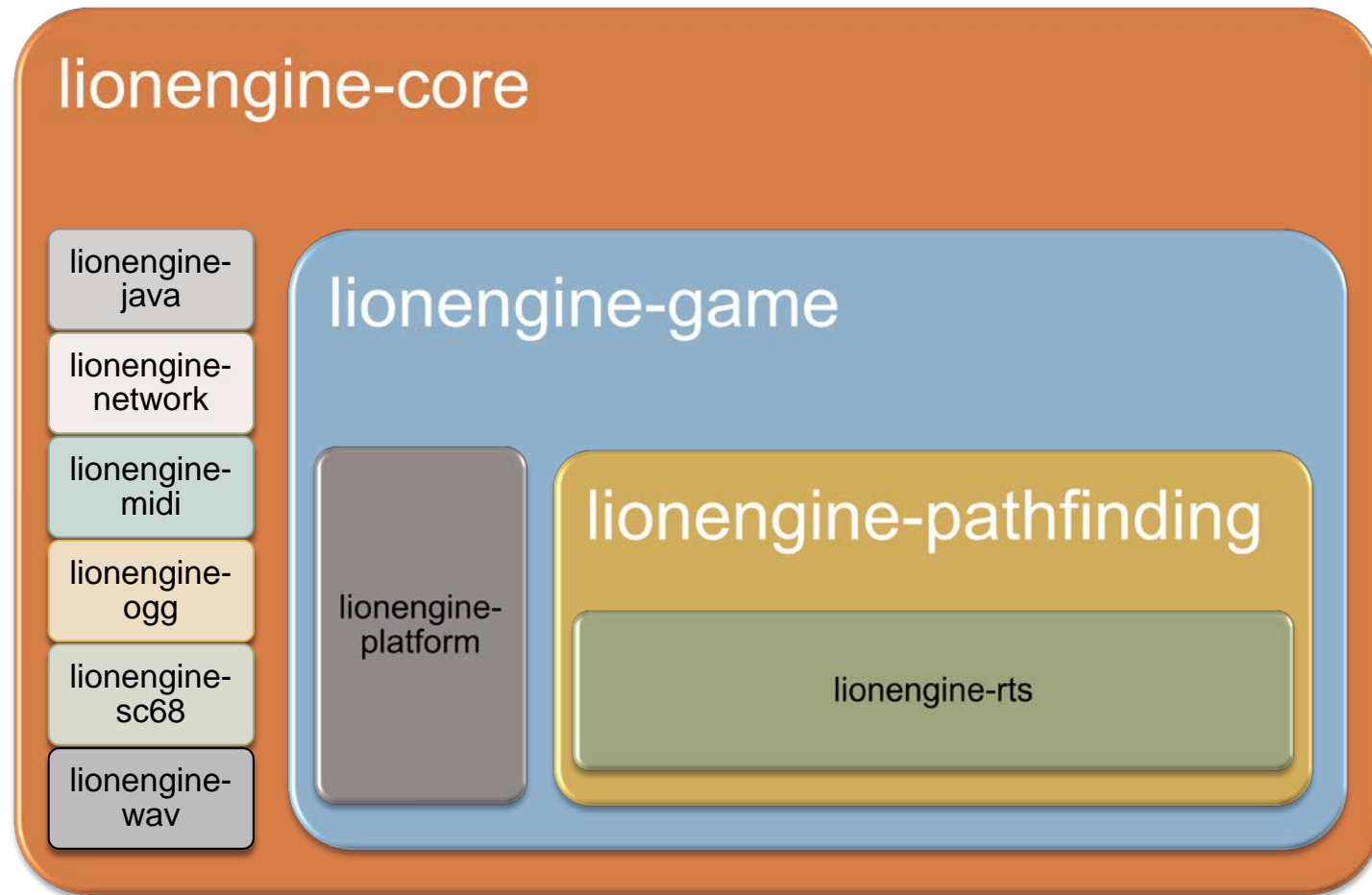
# Moteur - Modules

13

- Un module est présent sous la forme d'un JAR
  - ▣ Inclusion aisée des modules sur un projet
  - ▣ Nécessité d'inclure la partie centrale d'abord
  
- Chaque module
  - ▣ Dépend du module principal (`lionengine-core`)
  - ▣ Propose une base abstraite (`architecture de base`)
  - ▣ Est redéfinissable selon les besoins, en tout point
  - ▣ Est compatible avec d'autres modules
  - ▣ Respecte la même structure que la partie centrale

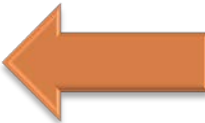
# Moteur - Modules

14



# Plan

15

- Moteur
- **Module Game** 
- Module Platform
- Module Rts
- Module Network

# Module Game

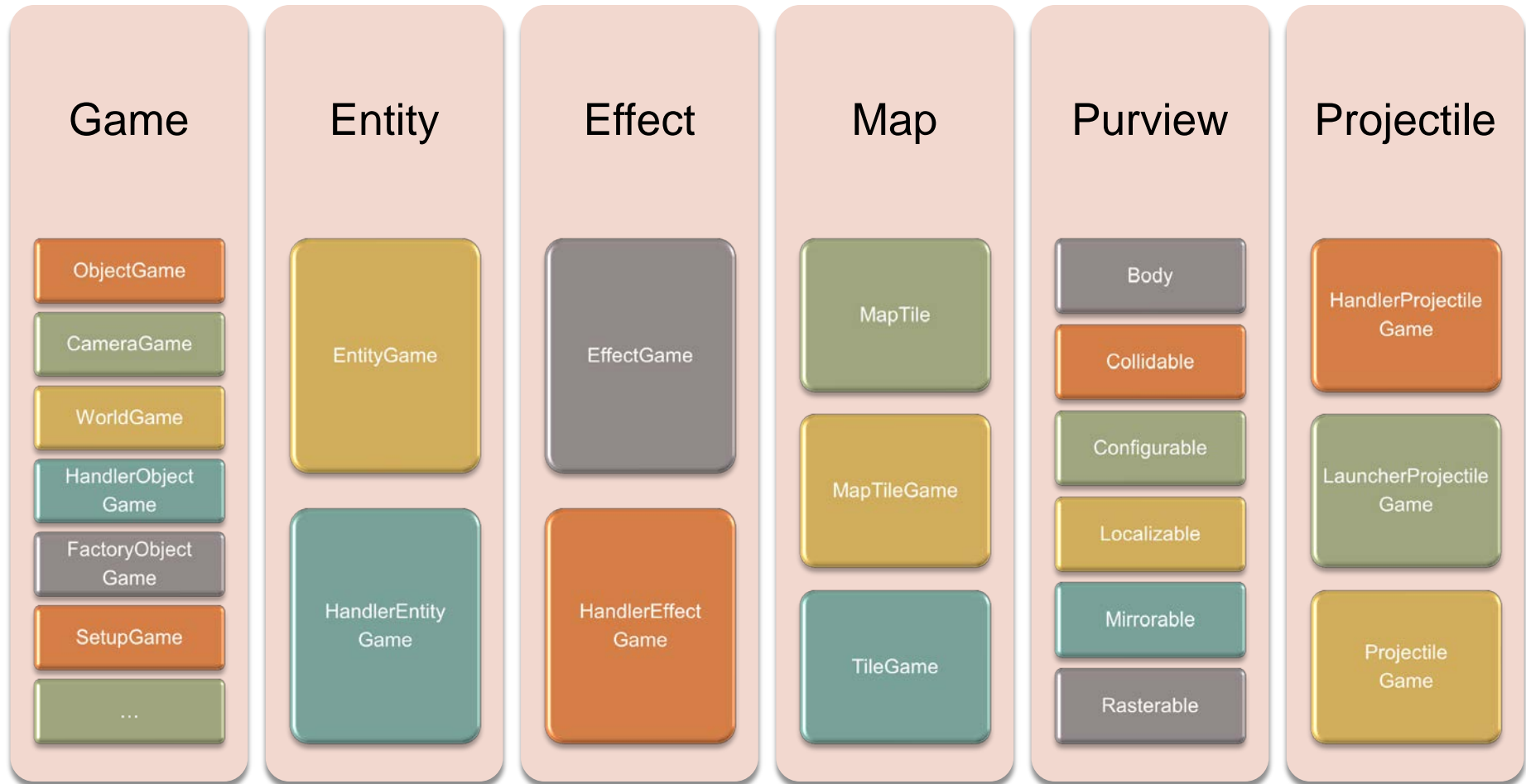
16

- Principal module abstrait
  - ▣ Sert de base dans le développement d'un jeu
  - ▣ Est utilisé par les modules plus spécifiques
    - Platform
    - Rts
    - Network
  - ▣ A besoin du moteur principal pour fonctionner



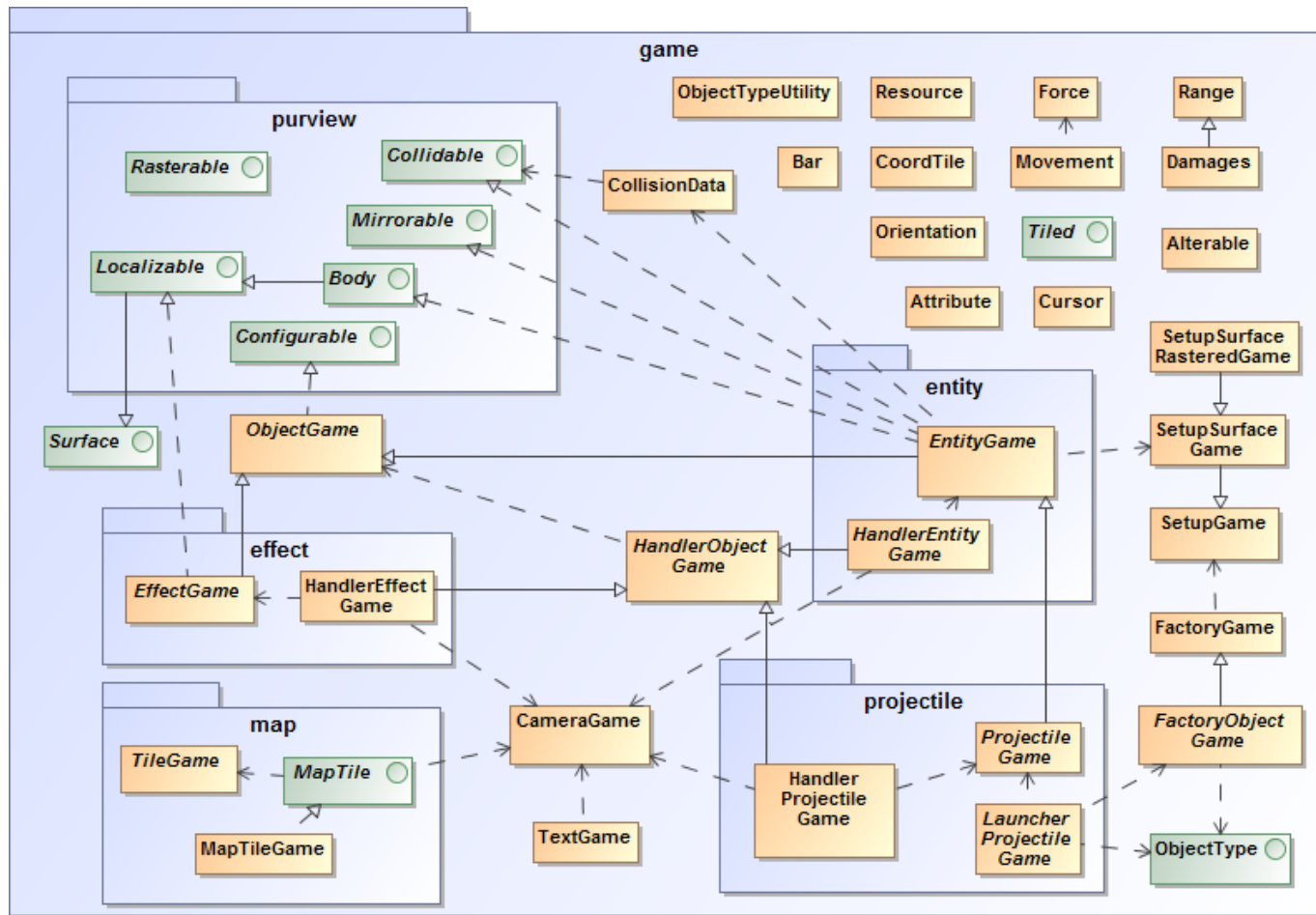
# Module Game - Structure

17



# Module Game - UML

18



# Module Game - Game

19

- Dans le package : `com.b3dgs.lionengine.game`
- Propose des types primaires
  - ▣ **ObjectGame** (représente un objet de base)
  - ▣ **CameraGame** (vue du joueur, évoluant dans le jeu)
  - ▣ **Damages** (gestion des dégâts, aléatoires ou non)
  - ▣ **Alterable** (facilite la manipulation de quantité)
  - ▣ **FactoryObjectGame** (chargé de créer les objets)
  - ▣ **HandlerObjectGame** (gère une collection d'objet)
  - ▣ **WorldGame** (conteneur: handler, map, camera...)
  - ▣ ...

# Module Game - Map

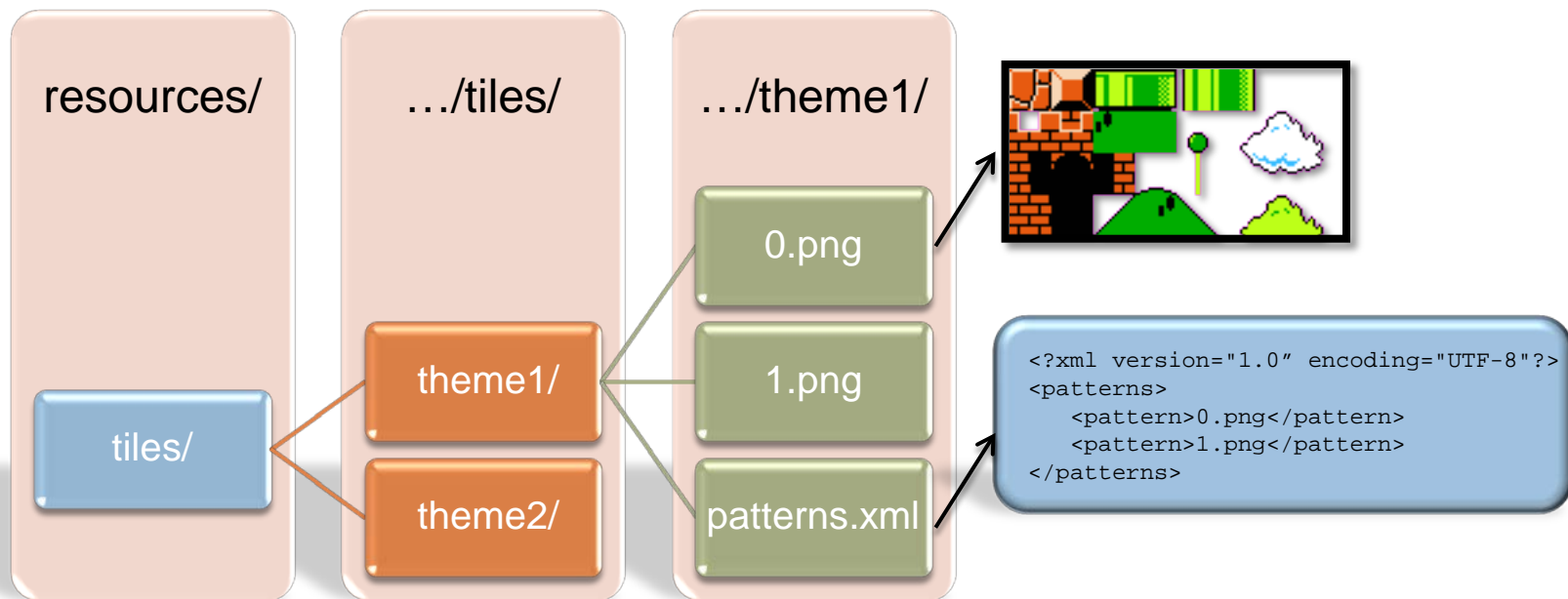
20

- Dans le package: `com.b3dgs.lionengine.game.map`
- Propose un type de map standard
  - ▣ MapTile (interface décrivant une map à base de tile)
    - MapTileGame (implémentation abstraite de base)
      - Chargement des tiles dans une image (SpriteTiled)
      - Import & export au format binaire
      - Associe les collisions aux tiles à partir d'un fichier externe
      - Génération d'une minimap représentant la map en pixel
    - TileGame (structure de base d'un tile)
      - Numéro de pattern (=N°image d'une tuile)
      - Numéro de tile (=N°tile dans la tuile)
      - Nom de la collision associée
      - Coordonnées sur la map (x, y)

# Module Game - Map

21

- L'architecture impose une structure de rangement
  - ▣ Un dossier, placé dans le dossier des ressources, doit contenir un dossier par thème, eux même contenant la liste des tilesheets avec le fichier 'patterns.xml'
  - ▣ Sans ce fichier, toutes les images de type .png seront chargées



# Module Game - Purview

22

- Dans le package: `com.b3dgs.lionengine.game.purview`
- Décrit des capacités supportées par des « Entity »
  - ▣ Collidable (gérant l'aspect collision de deux objets)
  - ▣ Configurable (configuration via un fichier xml)
  - ▣ Mirrorable (permet d'avoir son symétrique vertical)
  - ▣ Localizable (permet de gérer le placement d'un objet)
  - ▣ Rasterable (permet de gérer l'effet raster bar)
  - ▣ Body (représente un objet soumis à la gravité)

# Module Game - Utility

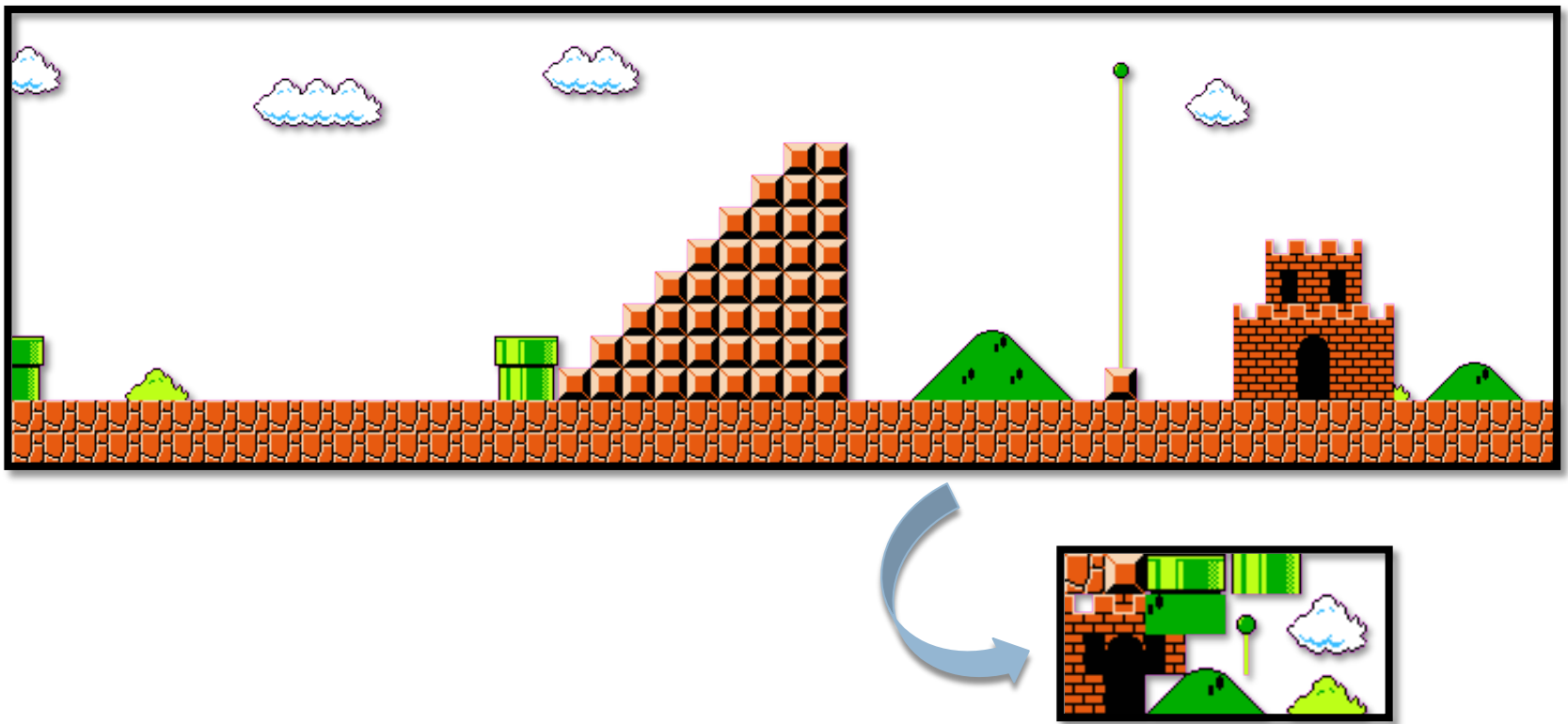
23

- Dans le package: `com.b3dgs.lionengine.game.utility`
- Conversion d'un « levelrip », en un format de donnée compatible MapTile
  - ▣ Charge un levelrip (image représentant un niveau entier)
  - ▣ Convertit au format MapTile
  - ▣ Sauvegarde au format binaire
  - ▣ Supporte le multithreading (meilleures performances)
- Extracteur de tile à partir d'un levelrip
  - ▣ Découpe les tiles uniques d'un levelrip et les sauvegarde dans une image en tilesheet

# Module Game - Utility

24

- *'TileExtractor'* en action:

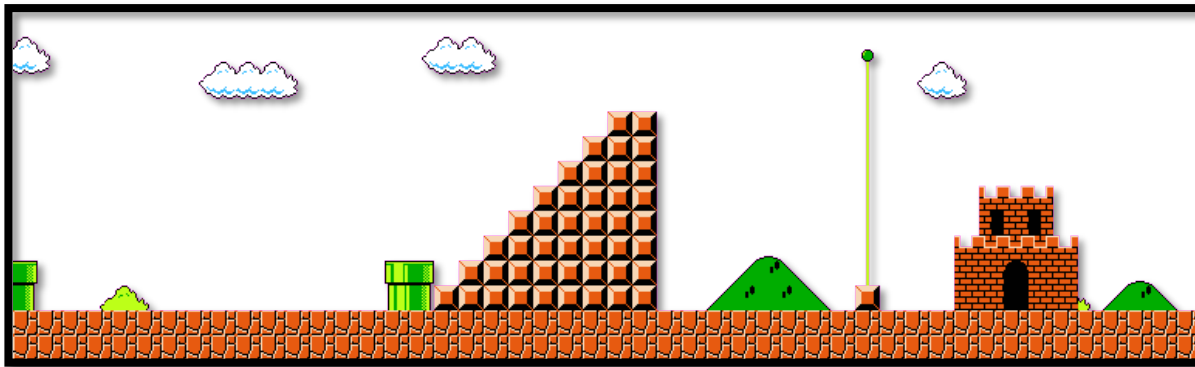




# Module Game - Utility

25


- *'LevelRipConverter'* en action:



MapTile  
(data of  
tiles)

# Plan

26

- Moteur
- Module Game
- **Module Platform** 
- Module Rts

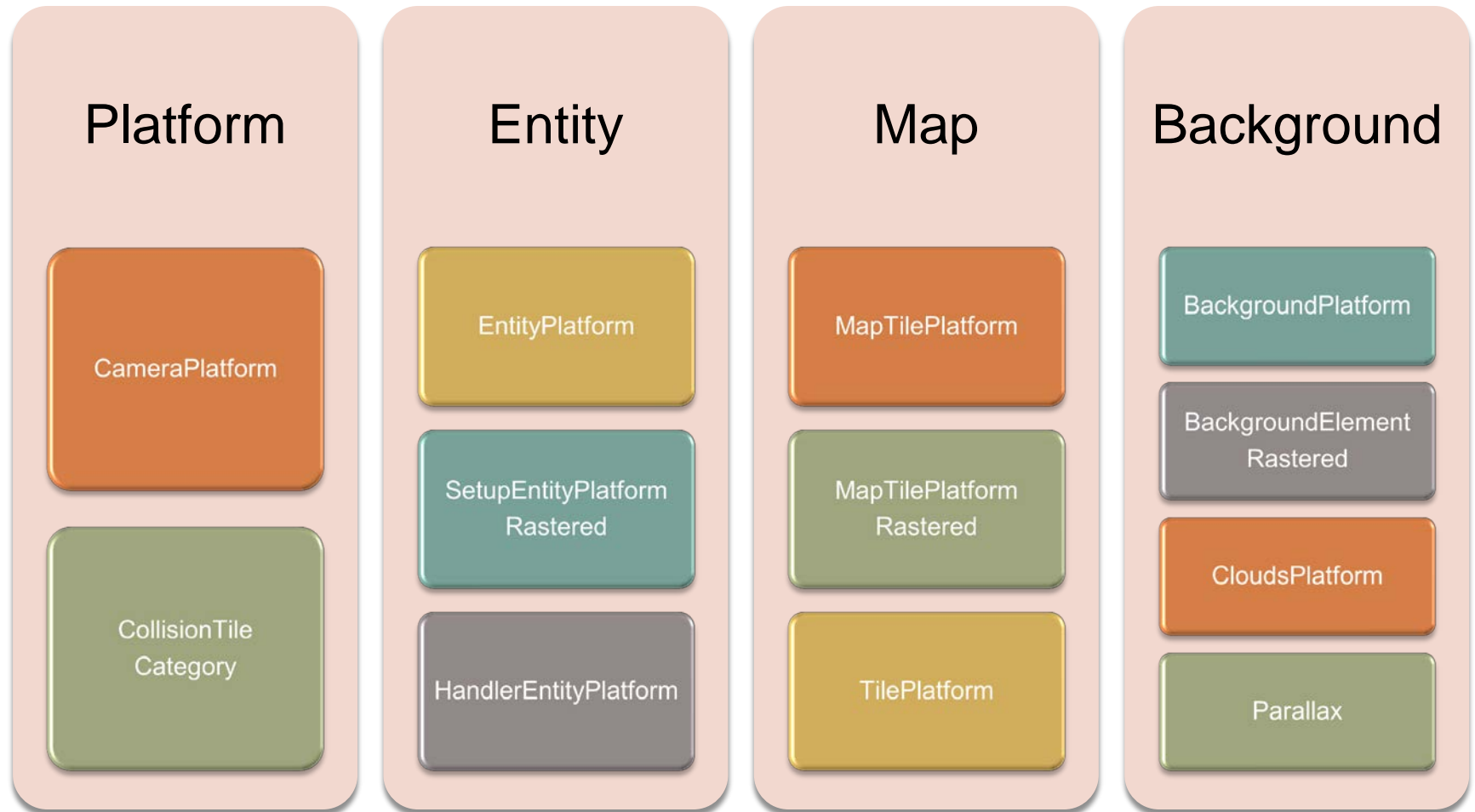
# Module Platform

27

- Module dédié aux jeux de plateforme
  - ▣ Propose des types de base
    - EntityPlatform
    - CameraPlatform
    - HandlerEntityPlatform
  - ▣ Permet une gestion plus avancée des maps
    - MapTilePlatformRastered
  - ▣ Contient un package complet dédié au background
    - Background, BackgroundRastered, Clouds, Parallax

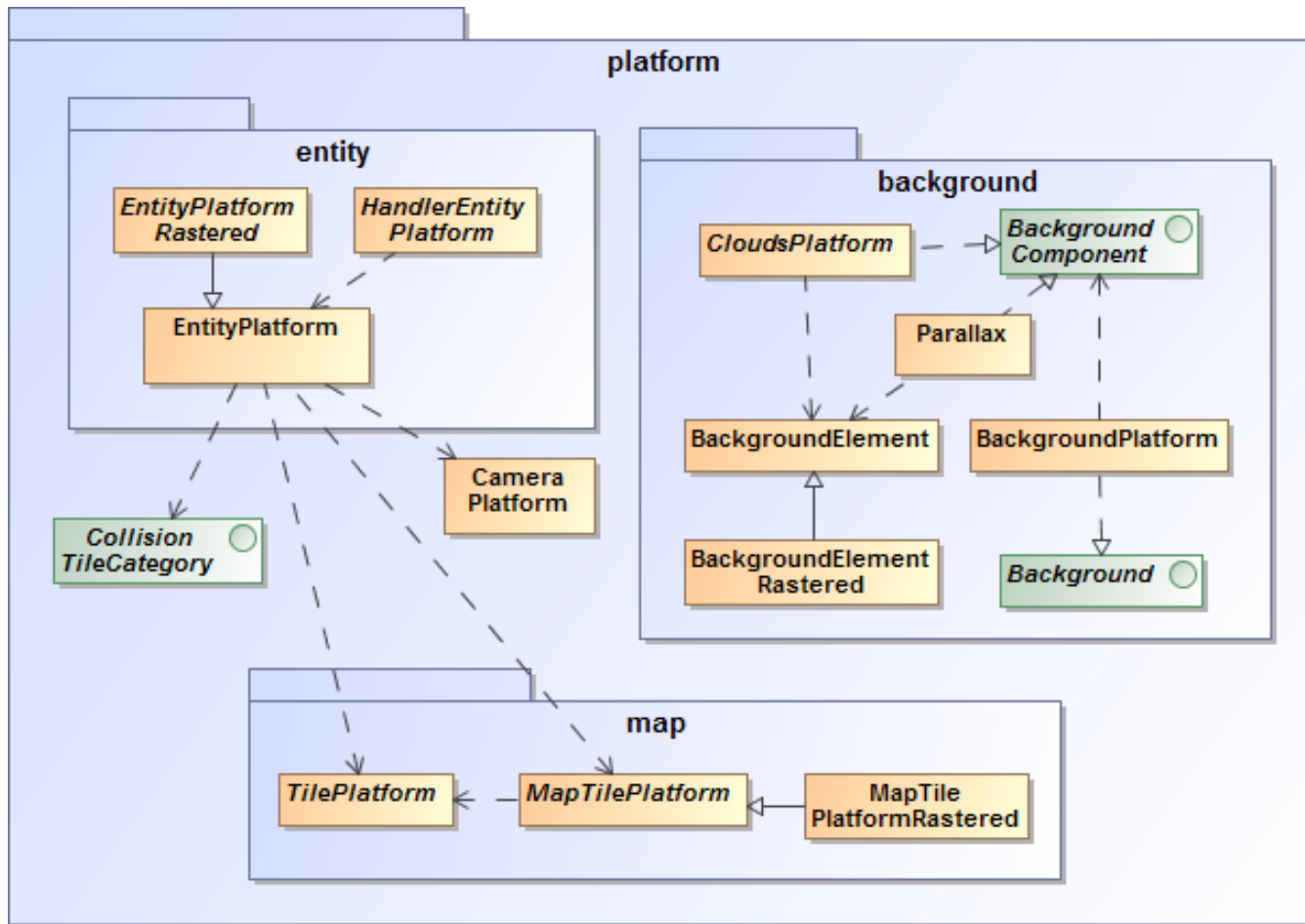
# Module Platform - Structure

28



# Module Platform - UML

29



# Module Platform - Package

30

- Dans le package: `com.b3dgs.lionengine.game.platform`
- Propose des types spécialisés « jeu de plateforme »
  - ▣ `EntityPlatform` (entité spécialisée)
  - ▣ `SetupEntityPlatformRastered` (partage des données)
  - ▣ `CameraPlatform` (caméra spécialisé)
  - ▣ `HandlerEntityPlatform` (handler spécialisé)
  - ▣ `BackgroundPlatform` (background orienté scrolling)
  - ▣ `Parallax` (effet 2.5D, basé sur une image)
  - ▣ `MapTilePlatform` (map spécialisée)

# Module Platform - Map

31

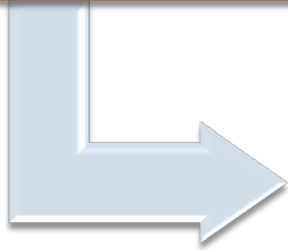
- La gestion des collisions est effectuée
  - ▣ En amont côté Tile
    - Récupère son type de collision depuis un fichier externe
    - Définit les points de collision en fonction d'une localisation
    - Dépend du type de collision du tile
  - ▣ En aval côté Entité
    - Teste quel est le premier tile traversé ayant une collision
    - Se place sur le point de collision du tile correspondant
    - Gère plusieurs collisions (sol, zones bloquantes...)

# Module Platform - Map

32

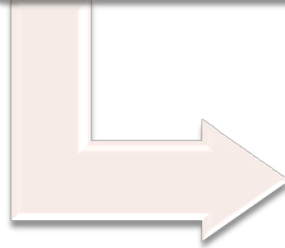
Check

- Recherche le premier tile ayant une collision suite à une intersection



Get

- Récupère le tile concerné, et recherche le point de collision associé



Apply

- L'entité se place sur le point de collision



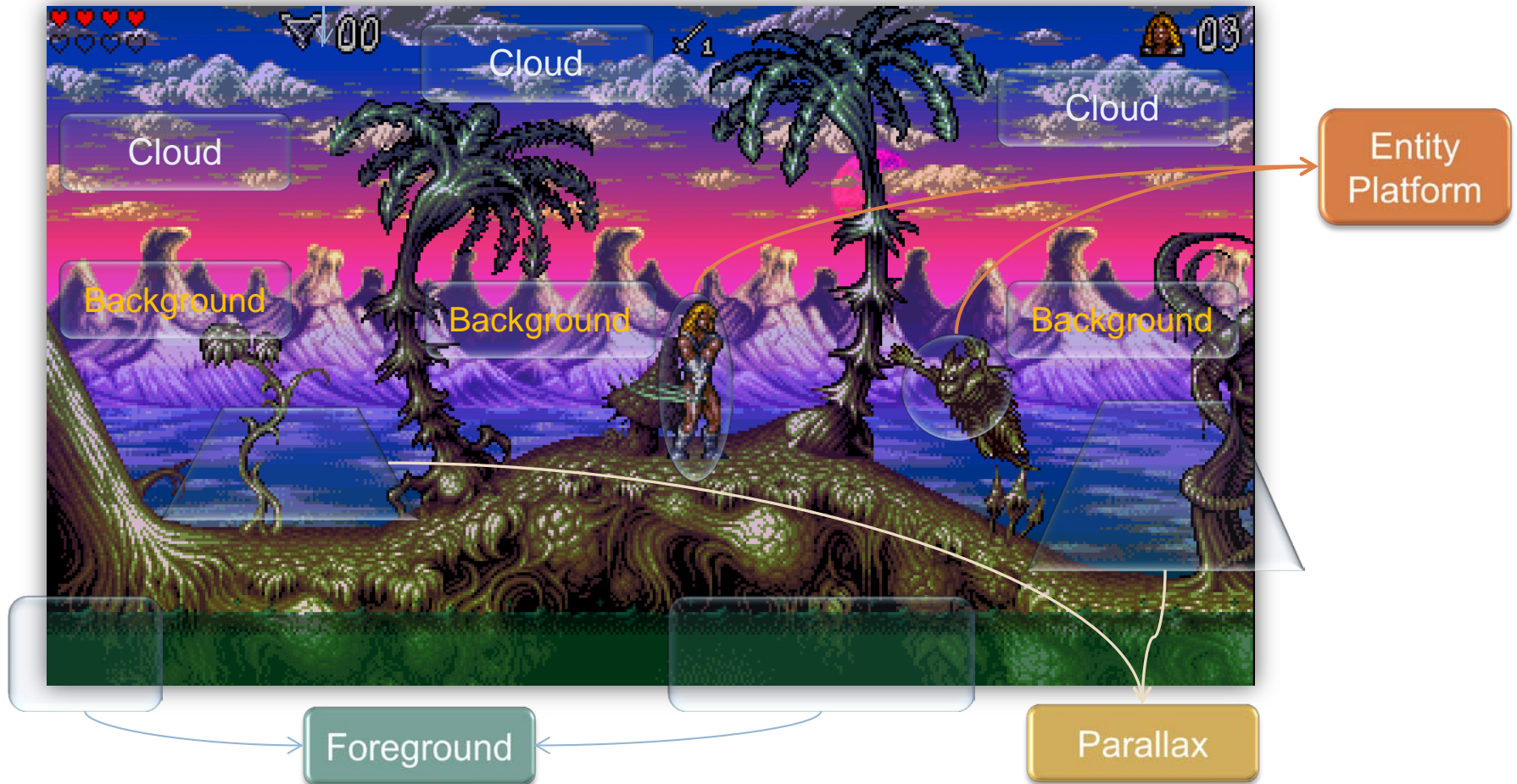
# Module Platform - Map

33

- `getCollisionTile` permet de récupérer le tile '*raycasté*'
  - ▣ Référentiel de l'entité (modulo un offset ajustable)
  - ▣ Possibilité de filtrer le type de collision recherché
- Récupération du point de collision du tile
  - ▣ L'appliquer à l'entité si il en existe un
- L'entité a récupéré le tile touché, et sa collision
  - ▣ Le processus est maintenant complet

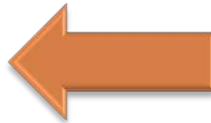
# Module Platform - Example

34



# Plan

35

- Moteur
- Module Game
- Module Platform
- **Module Rts** 
- Module Network

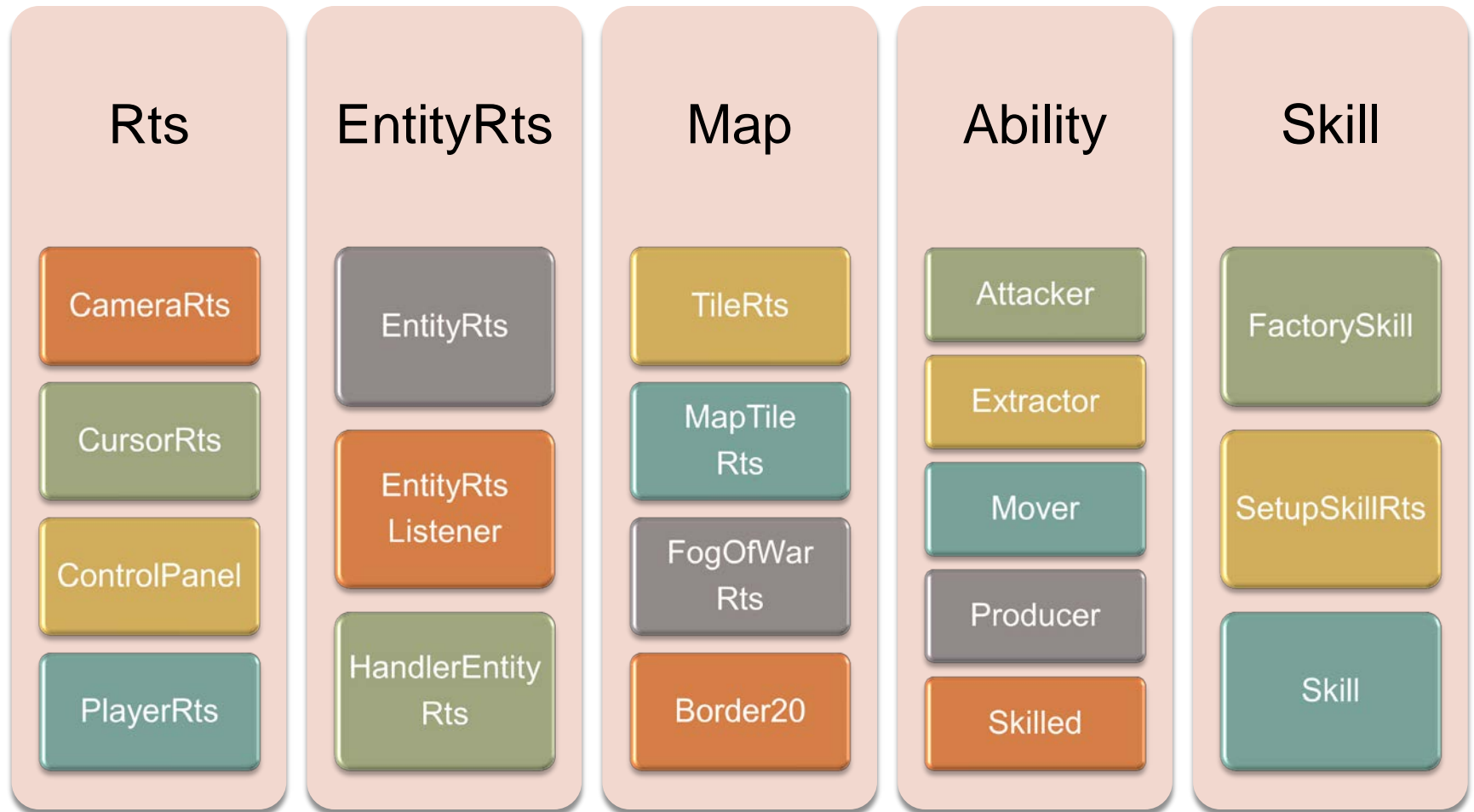
# Module Rts

36

- Module dédié aux jeux de stratégie
  - ▣ Propose des types de base
    - EntityRts
    - CameraRts
    - CursorRts
    - SkillRts
  - ▣ Permet une gestion plus avancée des maps
    - MapTileRts
  - ▣ Contient un package complet dédié au pathfinding

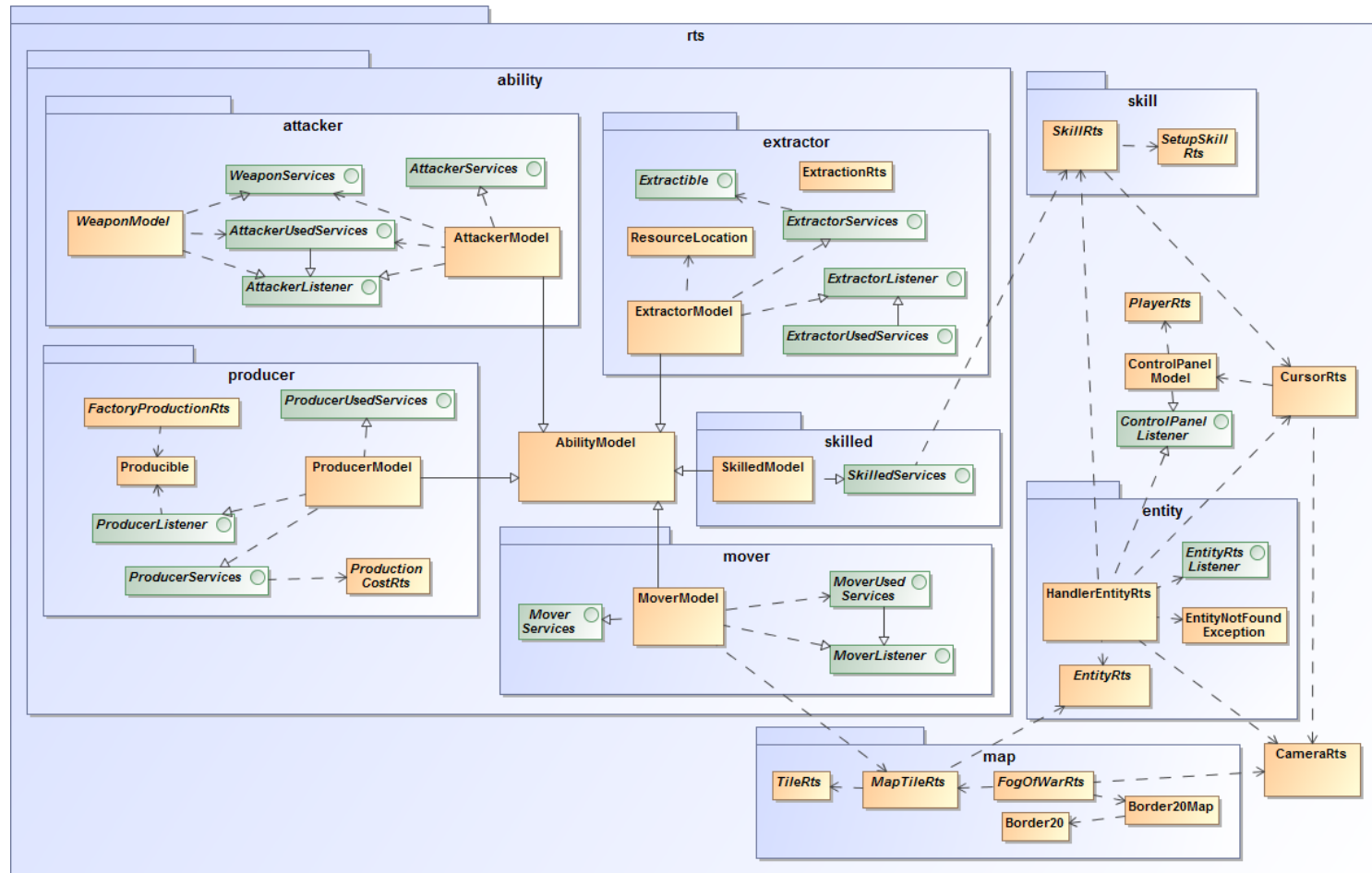
# Module Rts - Structure

37



# Module Rts - UML

38



# Module Rts - Package

39

- Dans le package: `com.b3dgs.lionengine.game.rts`
- Propose des types spécialisés « jeu de stratégie »
  - ▣ `EntityRts` (entité spécialisé)
  - ▣ `CameraRts` (caméra spécialisé)
  - ▣ `ControlPanel` (panneau de contrôle)
  - ▣ `HandlerEntityRts` (handler spécialisé)
  - ▣ `SkillRts` (base abstraite d'une compétence)
  - ▣ `PlayerRts` (représentation de base d'un joueur)

# Module Rts - Exemple

40

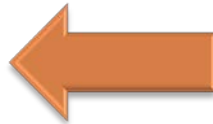




# Plan

41

- Moteur
- Module Game
- Module Platform
- Module Rts
- **Module Network**



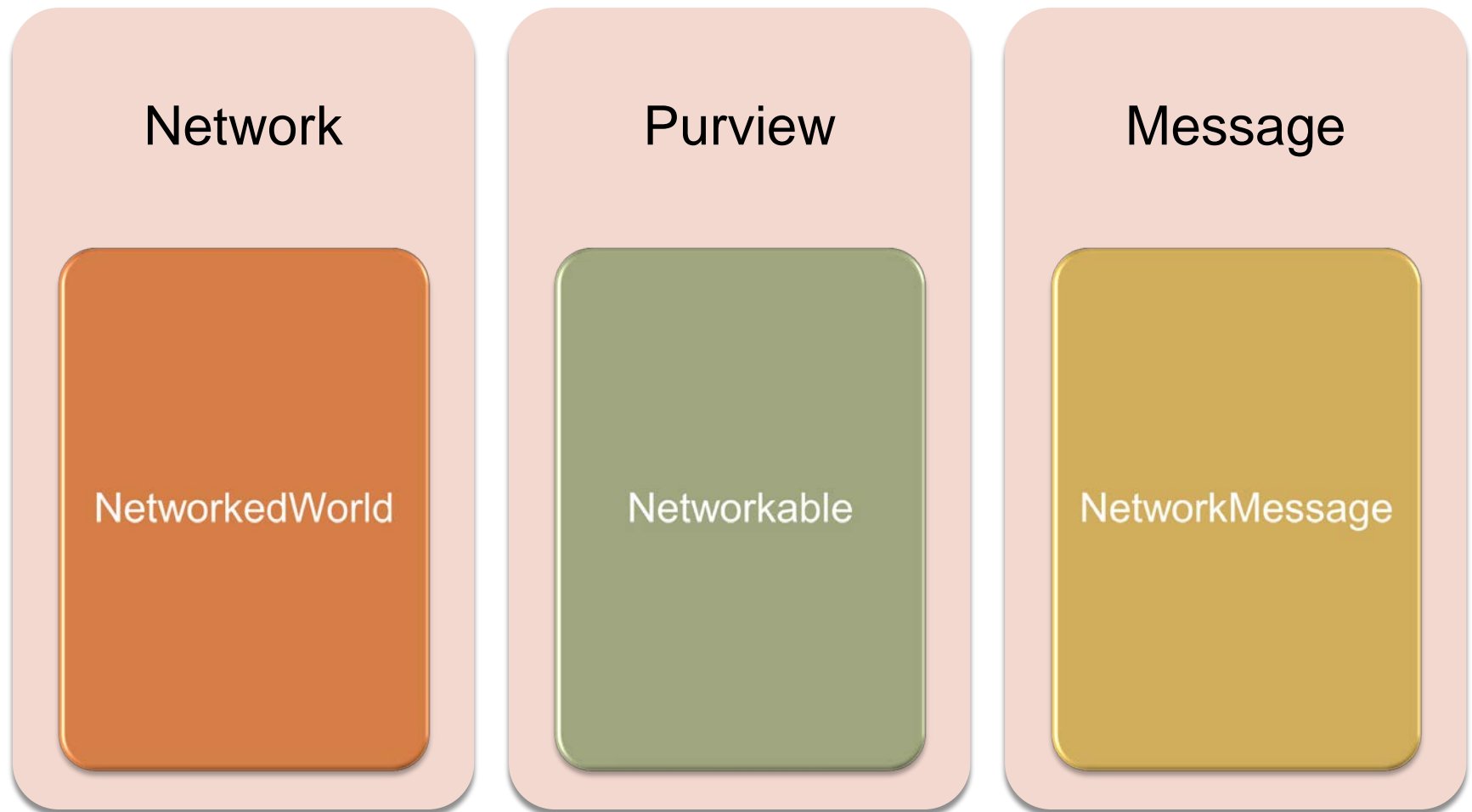
# Module Network

42

- Module dédié aux jeux en réseau
  - ▣ Propose des types de base

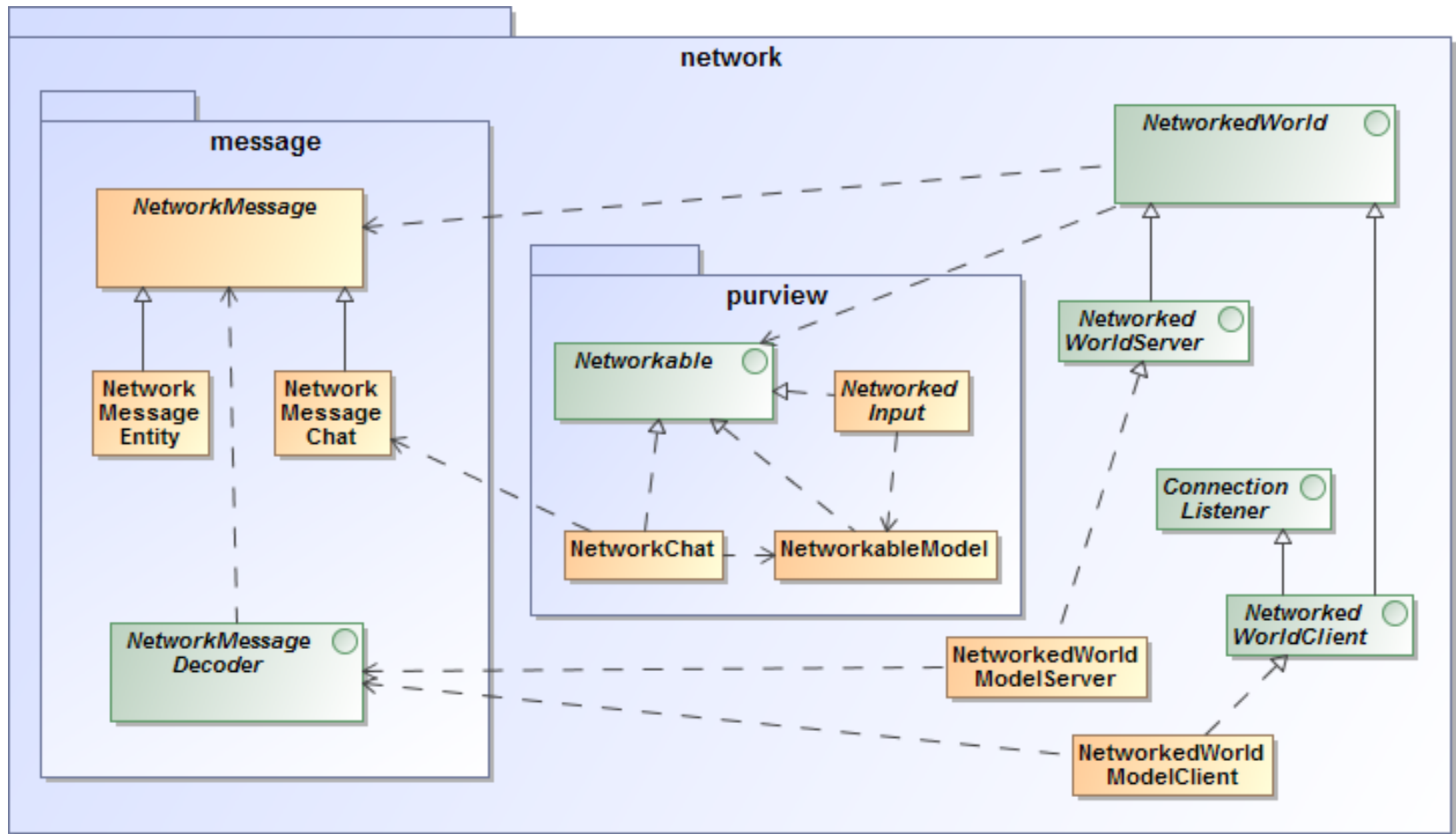
# Module Network - Structure

43



# Module Network - UML

44



# Module Network - Package

45

- Dans le package: `com.b3dgs.lionengine.network`
- Propose des types spécialisés « network »

# Module Network - Exemple

46

