Soutenance Module 1 LUA

Comment jouer ?

Le but du jeu

Le but du jeu est simple, chaque joueurs (joueur humain et ordinateur) possèdent une tour principale surmontée d'une tourelle et protégée par un bouclier.

Tant que le bouclier est actif il est impossible de faire des dégâts à une tourelle principale.
Pour faire tomber le bouclier d'une tourelle principale il faut capturer des tours secondaires placées à différents endroits de la carte.

Quand un joueur (humain ou ordinateur) possède le contrôle de toutes les tours secondaires le bouclier de la tour principale de son adversaire est désactivé.
Pour prendre le contrôle d'une tour secondaire il suffit de l'attaquer, quand elle n'a plus de vie elle change de propriétaire et regagne son maximum de points de vie.

Comment contrôler le tank ?

Pour déplacer le tank sur la carte, il faut utiliser les touches **ZQSD du clavier**.
Pour tirer sur une cible, il faut utiliser le **bouton gauche de la souris en visant un endroit sur la carte avec la souris** (la tourelle du tank tourne a 360 dégrées).
Plus on tire loin, plus la vitesse du tir est réduite, plus tire une cible proche plus la vitesse du tir est rapide.

Comment gagner ou comment perdre ?

Pour gagner il faut détruire la tour principale de l'adversaire, pour perdre il faut laisser l'ordinateur détruire notre tour principale.

Comment j'ai fait ?

Pour la réalisation de ce jeu, j'utilise un système de composants (Component)

Pour la définition des composants j'ai utilisé l'atomic design adapté aux composants de jeu vidéo.

Combien de temps pour le projet ?

43 heures

Qu'avez-vous appris ?

Love et LUA, que je ne connaissais pas du tout avant.

Quelles sont les difficultés que vous avez rencontrées ?

La gestion des classes, l'héritage dans LUA qui n'est pas aussi poussé qu'en C#.

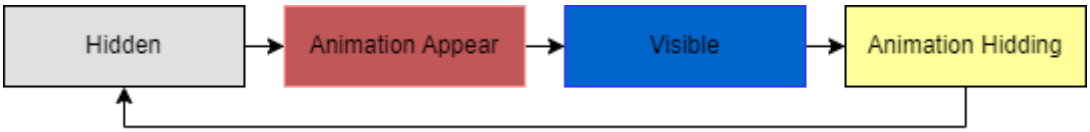
Quels sont les domaines dans lesquels vous pensez devoir vous améliorer

La gestion du brouillard de guerre pourrait être mieux, j'aimerai utiliser des draw-arc plutôt que de simple carrés noirs. La gestion des collisions est très basique et parfois frustrante en jeu, il faudrait plutôt faire des colliders ronds.

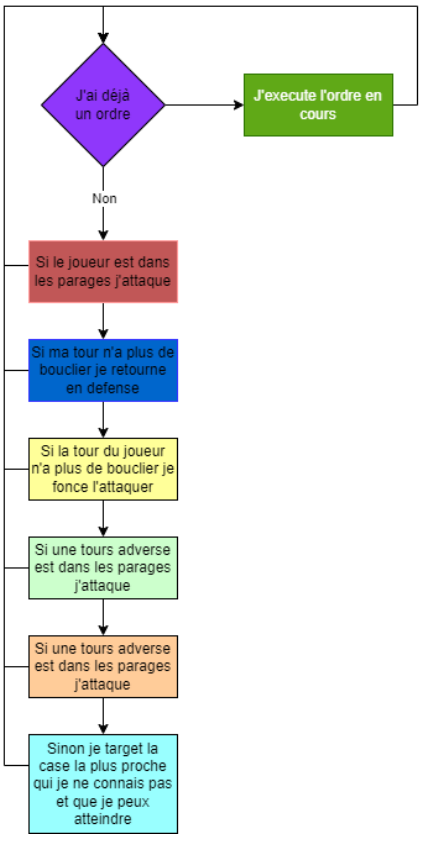
Machine à état ?

Dans le projet il existe plusieurs composants qui utilisent des machines à état plus ou moins difficiles pour exister.

L'animation des frames :



Le choix de l'attitude de l'ennemi :



Pour la réalisation de ce jeu, j'utilise un système de composants (Component)

Pour la définition des composants j'ai utilisé l'atomic design adapté aux composants de jeu vidéo.

L'idée était donc de développer des composants (atomes) (images, text, sons, etc.) qui peuvent être combinés pour créer des composants (molécules) plus complexes (bouton, menu, etc.) pour ensuite les combiner à nouveau dans des éléments plus complexes du jeu (organismes) (joueur, ennemi, interface, etc.) pour finalement les combiner dans des scenes (templates, pages)

Les bases de l'architecture

Les classes de bases de l'architecture sont 5 classes qui gèrent les interactions entre love et le framework du jeu

---

Les composants du framework

Les atomes

Les atomes de base du framework se trouvent dans le répertoire /framework/…

Les atomes sont les composants de base, ils ne contiennent pas d'autres composants

---

Les molécules

Les molécules de base du framework se trouvent dans le répertoire /framework/…

Les molécules utilisent les atomes pour créer des composants plus complexes

---

Les organismes du jeu

Les molécules de base du framework se trouvent dans le répertoire /models/…

Les organismes sont des composants plus complexes qui contiennent des molécules et des atomes

---

Les templates/scenes du jeu

Les templates/scenes de base du framework se trouvent dans le répertoire /scenes/…

Les templates/scenes sont des composants plus complexes qui contiennent des organismes, des molécules et des atomes

---

Les classes utilitaires

Les classes de base du framework se trouvent dans le répertoire /framework/tools et /framework/drawing

Les classes utilitaires sont des classes qui apportent des fonctionnalités partagées par les composants du framework

Les bases de l'architecture

Les classes de bases de l'architecture sont 5 classes qui gèrent les interactions entre love et le framework du jeu

framework/configuration/Configuration.lua

La classe configuration enregistre dans un fichier les éléments de configuration de l'utilisateur (full-screen, vsync, volume de la musique, volume des effets sonores, niveau de difficulté, retourne également les différents paramètres du jeu en fonction de la difficulté)

framework/screen/ScreenManager.lua

ScreenManager permet de gérer les changements de résolution, lors de son initialisation l'objet ScreenManager est initialisé avec la résolution de développement. Ensuite la classe permet d'obtenir le ratio à appliquer à chaque composant du jeu afin de s'adapter à la résolution actuelle.

Le cœur de l'architecture en composants

framework/scenes/ScenesManager.lua

ScenesManager gère la collection de scène actuellement actives dans le jeu, il permet également d'appeler automatiquement les méthodes load, update, draw, unload de chaque scène.

framework/scenes/Scene.lua

La classe Scene est une classe abstraite dont chaque scène hérite, les scenes contiennent une collection de composants, c'est la scène qui a la charge d'appeler les méthode load, update, draw, unload de chaque composant qu'elle contient.

framework/scenes/Component.lua

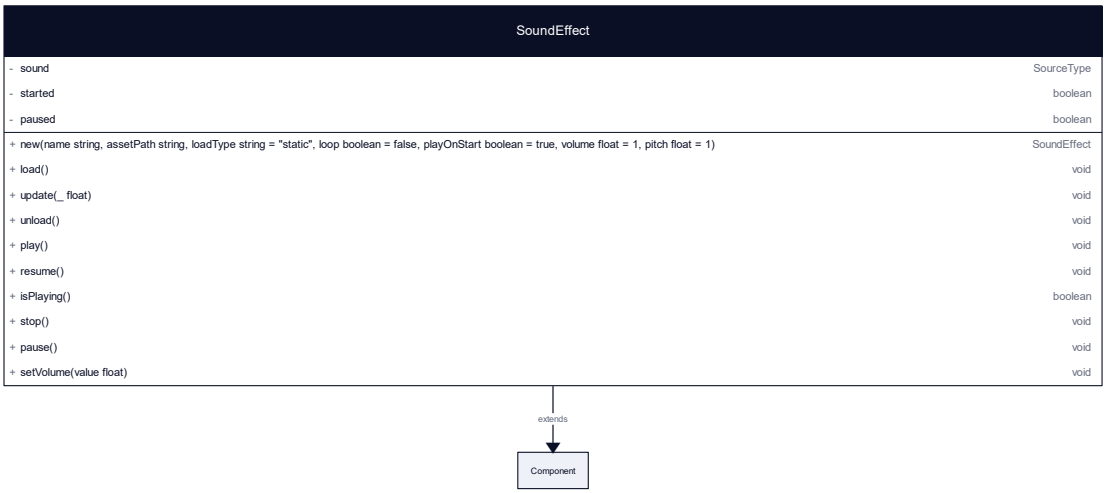
La classe Component est une classe abstraite dont chaque composant hérite, les composants contiennent une collection de sous-composants, c'est le composant qui a la charge d'appeler les méthodes load, update, draw, unload de chaque sous-composant qu'il contient.

Les atomes

Les atomes sont les composants de base, ils ne contiennent pas d'autres composants

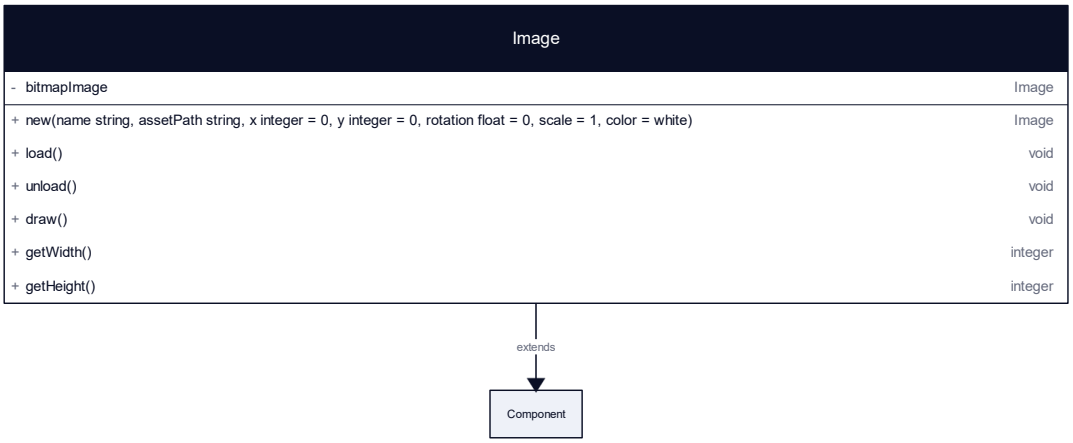
framework/audio/SoundEffect.lua

Le composant SoundEffect permet de lire un fichier audio, il possède des méthodes play, stop, pause



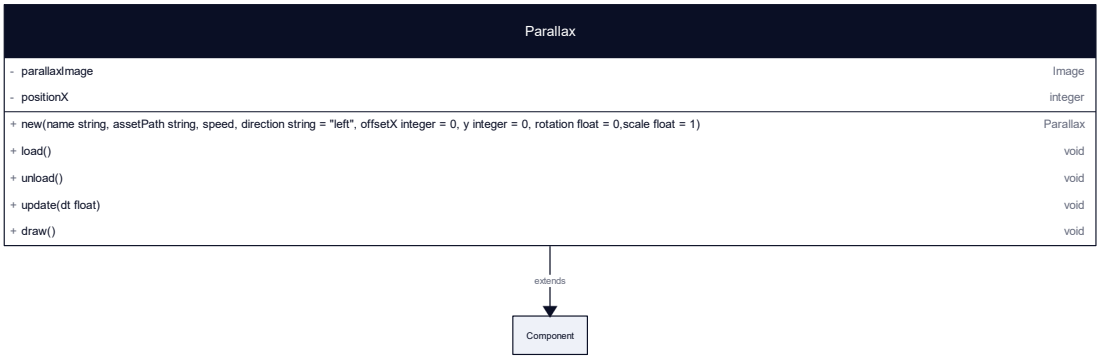
framework/images/Image.lua

Le composant Image permet d'afficher une image, il possède des propriétés permettant de changer l'orientation, la taille



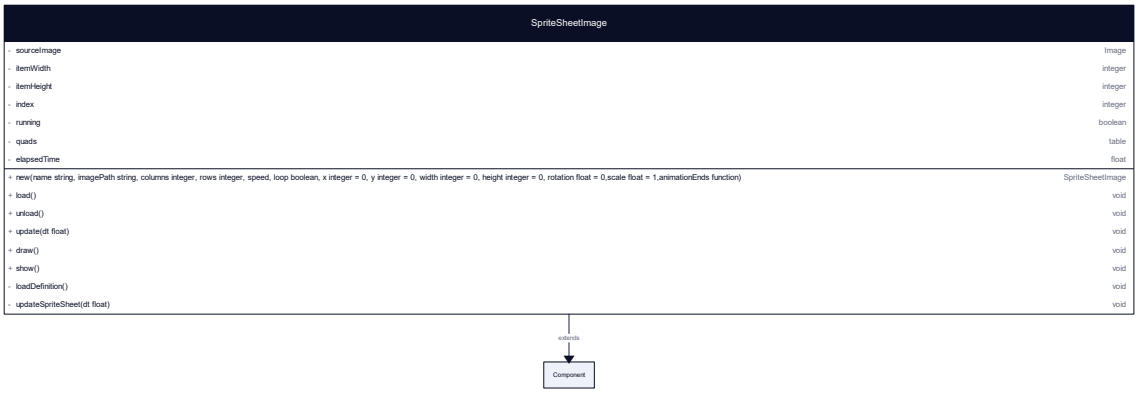
framework/images/Parallax.lua

Le composant Parallax permet d'afficher une image en appliquant un effet de mouvement sur l'axe X



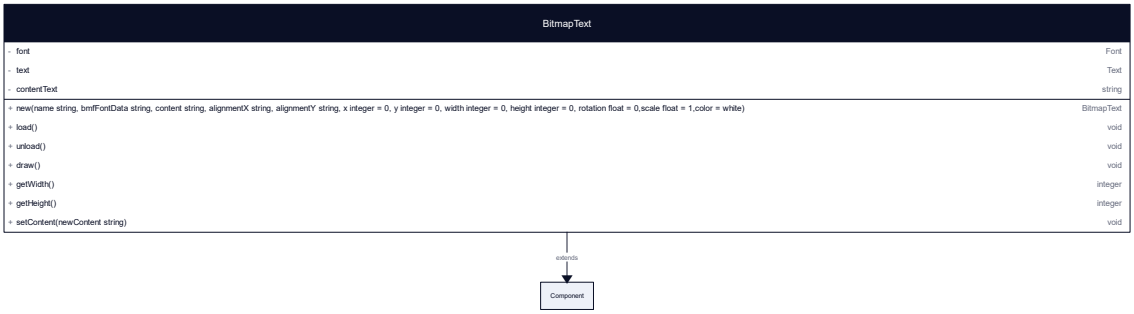
framework/images/SpriteSheetImage.lua

Le composant SpriteSheetImage permet d'afficher un sprite sheet animé en définissant le nombre de colonnes, de lignes, la vitesse de lecture



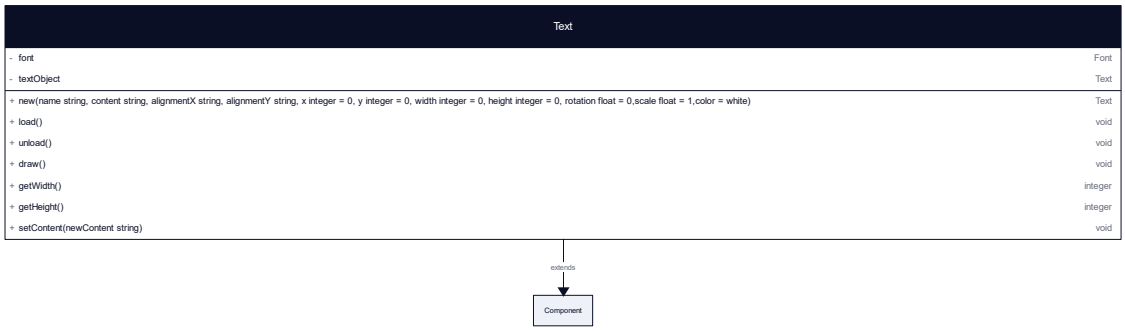
framework/texts/BitmapText.lua

Le composant BitmapText permet d'afficher un texte en utilisant une bitmap font



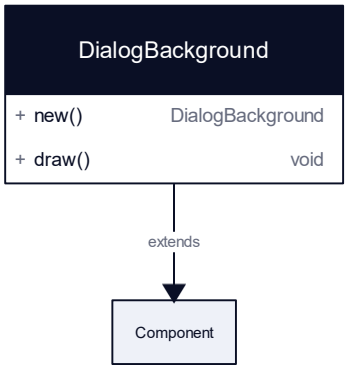
framework/texts/Text.lua

Le composant Text permet d'afficher un texte en utilisant la font standard de love



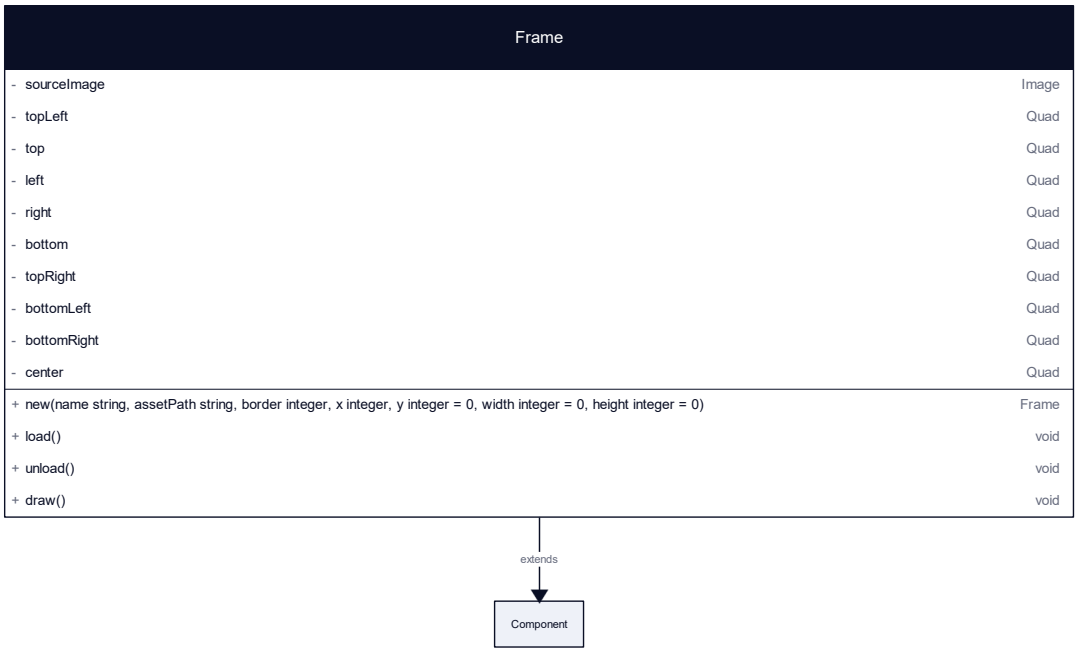
framework/ui/DialogBackground.lua

Le composant DialogBackground permet d'afficher un fond semi-transparent pour les dialogues et les menus



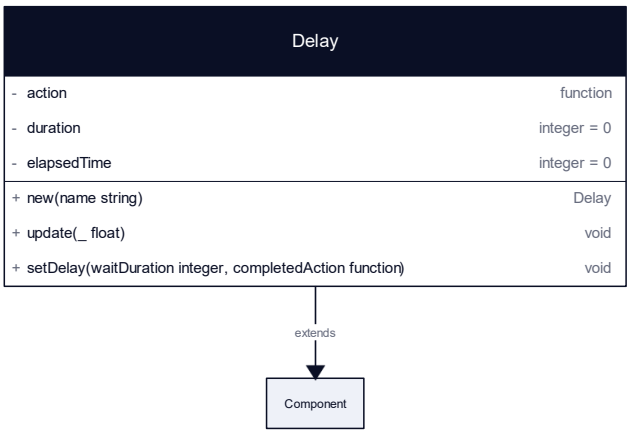
framework/ui/Frame.lua

Le composant Frame permet d'afficher une fenêtre en utilisant une texture compatible nine-patch



framework/tools/Delay.lua

Le composant Delay permet de déclencher un callback après un certain délai

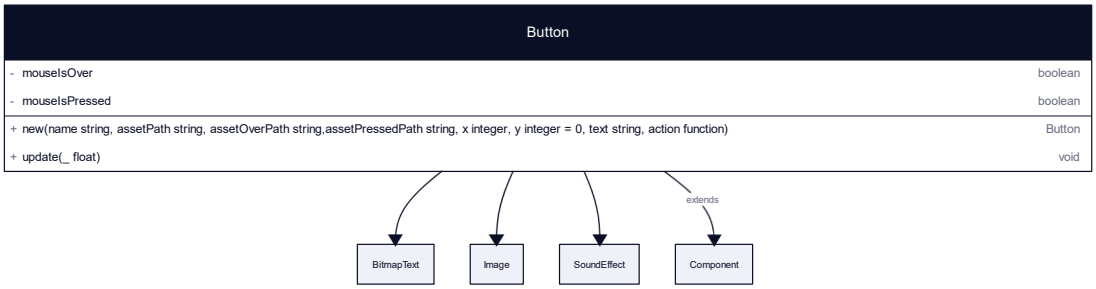


Les molécules

Le molécules utilisent les atomes pour créer des composants plus complexes

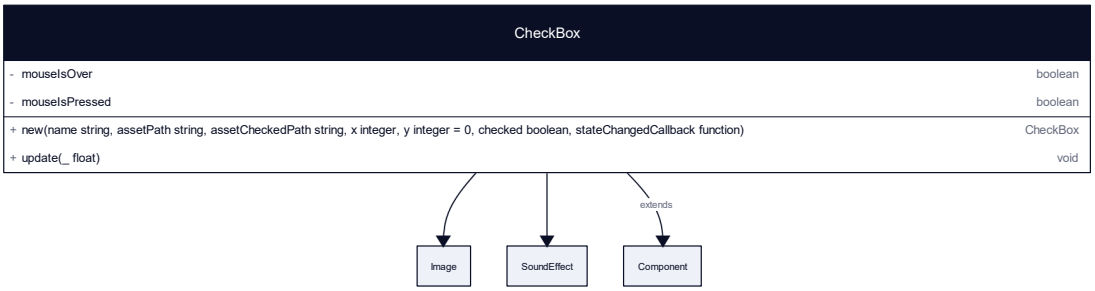
framework/ui/Button.lua

Le composant Button permet d'afficher un bouton avec l'effet au survol, cliqué, il permet également d'exécuter un callback lors du click



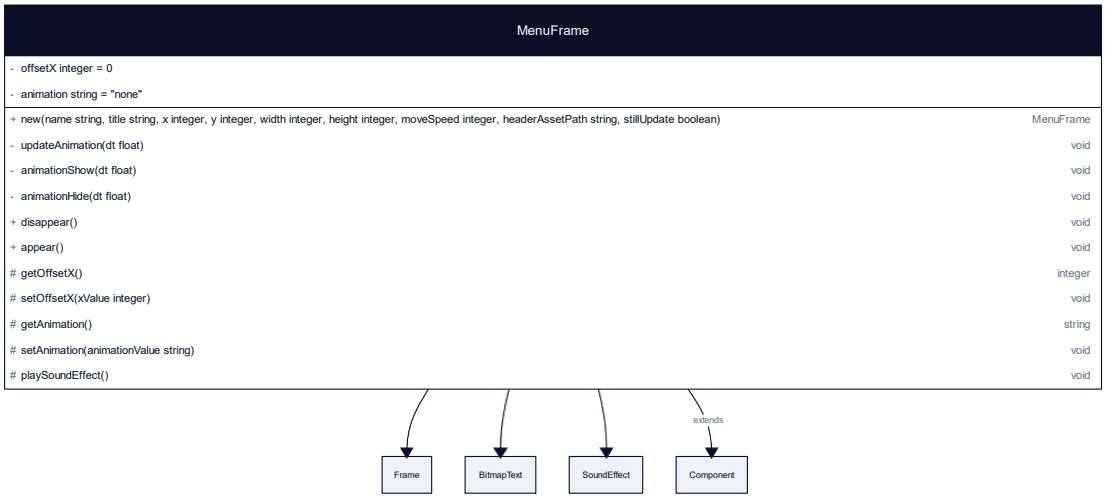
framework/ui/CheckBox.lua

Le composant CheckBox permet d'afficher une checkbox



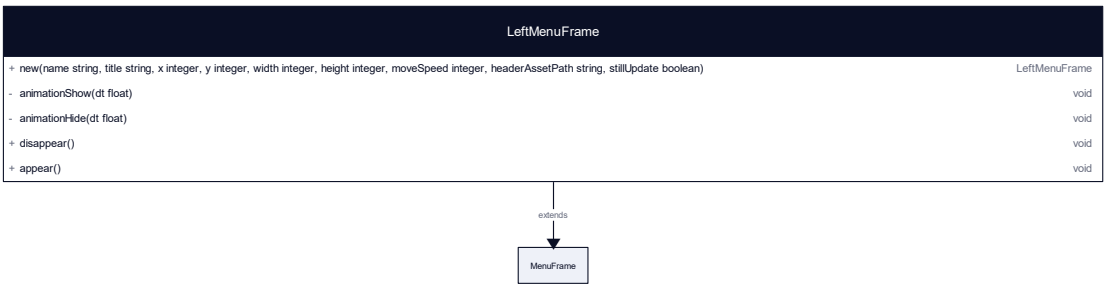
framework/ui/MenuFrame.lua

Le composant MenuFrame permet d'afficher une fenêtre qui arrive depuis la droite en utilisant une texture compatible nine-patch



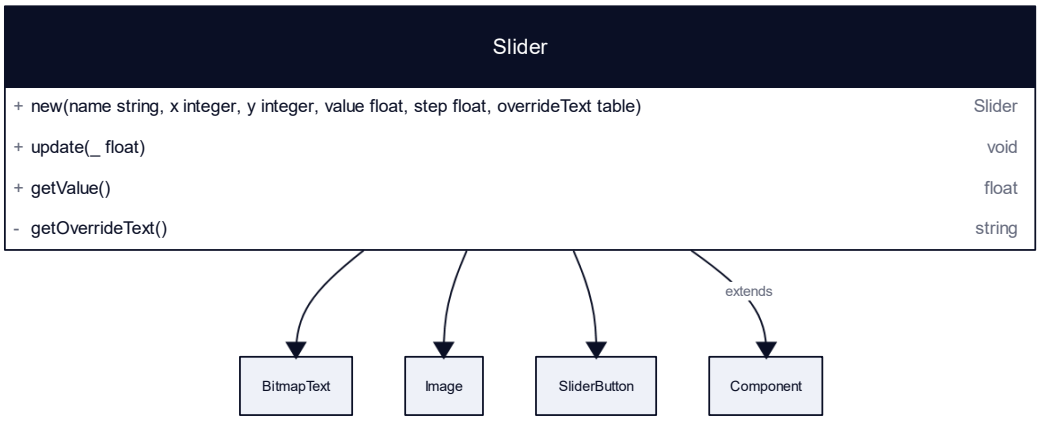
framework/ui/LeftMenuFrame.lua

Le composant LeftMenuFrame permet d'afficher une fenêtre qui arrive depuis la gauche en utilisant une texture compatible nine-patch



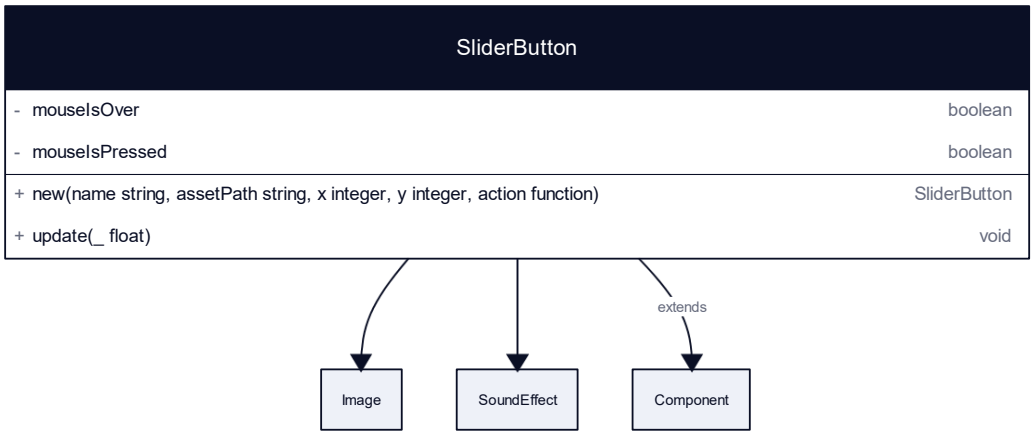
framework/ui/Slider.lua

Le composant Slider permet d'afficher un slider qui permet à l'utilisateur de sélectionner une valeur comprise entre 0 et 1



framework/ui/SliderButton.lua

Le composa nt SliderButton est utilisé dans le composant Slider afin d'afficher les boutons + et - qui permettent de modifier la valeur de ce-dernier.



Les organismes du jeu

Les organismes sont des composants plus complexes qui contiennent des molécules et des atomes

J'ai séparé les organismes en 3 catégories en fonction des scenes du jeu :

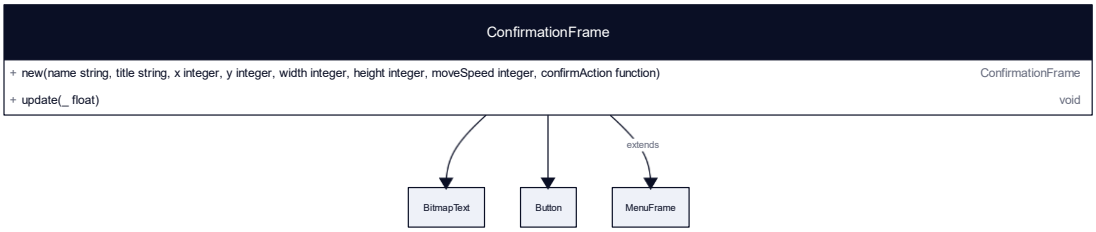
- Menu principal
- Sélection de niveau
- Niveau du jeu

Les organismes de la scène : Menu principal

Les organismes de la scène Menu principal sont les organismes qui composent la scène du menu principal du jeu.

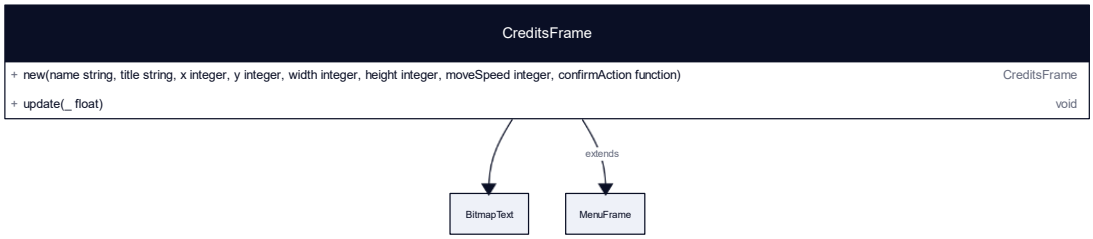
models/mainmenu/ConfirmationFrame.lua

Le composant ConfirmationFrame est l'écran de confirmation affichée lorsque l'utilisateur souhaites quitter le jeu



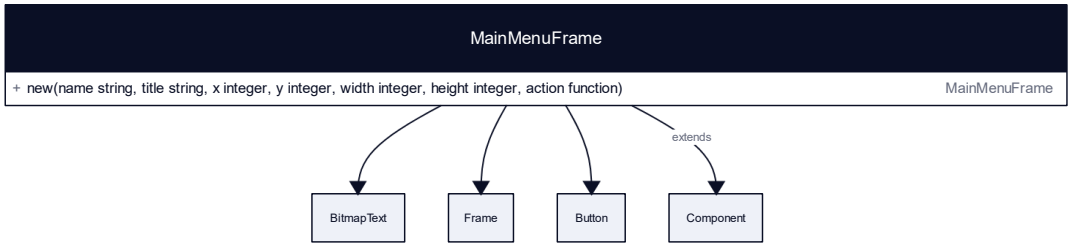
models/mainmenu/CreditsFrame.lua

Le composant CreditsFrame est l'écran qui s'affiche pour indiquer les crédits du jeu



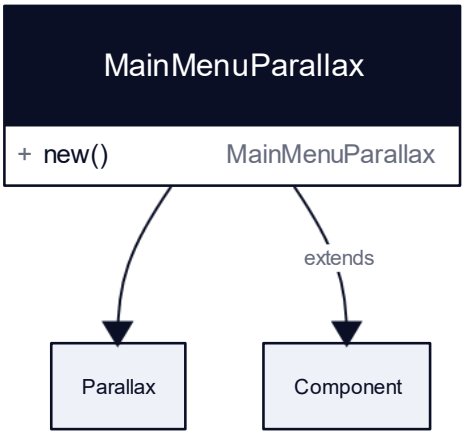
models/mainmenu/MainMenuFrame.lua

Le composant MainMenuFrame est la frame qui contient le menu principal du jeu



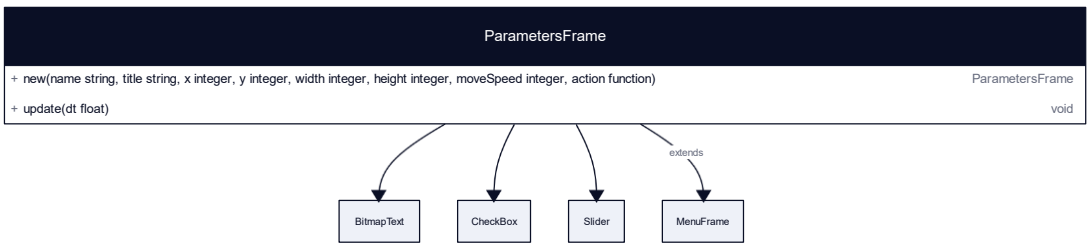
models/mainmenu/MainMenuParallax.lua

Le composant MainMenuParallax permet d'afficher l'ensemble des parallax de la scène du menu principal



models/mainmenu/ParametersFrame.lua

Le composant ParametersFrame est l'écran qui s'affiche pour afficher les paramètres du jeu

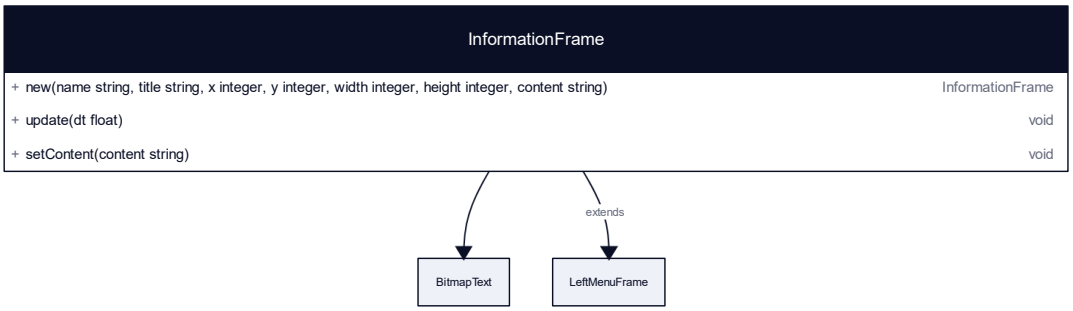


Les organismes de la scène : Sélection de niveau

Les organismes de la scène Sélection de niveau sont les organismes qui composent la scène de sélection du niveau de jeu.

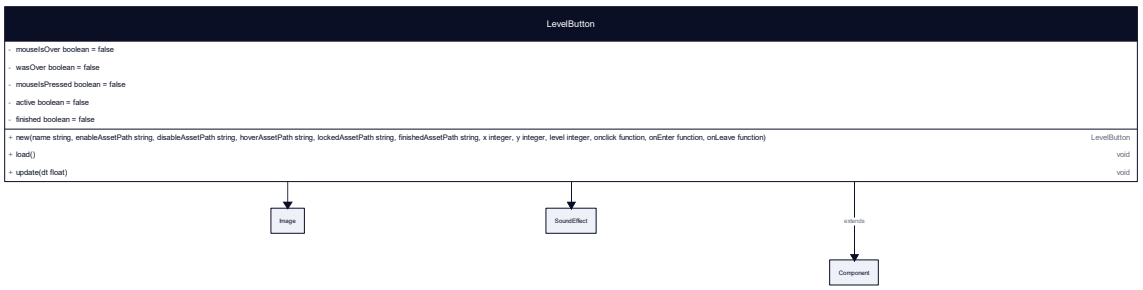
models/levelselect/InformationFrame.lua

Le composant InformationFrame est l'écran qui affiche les informations du niveau au survol de la souris



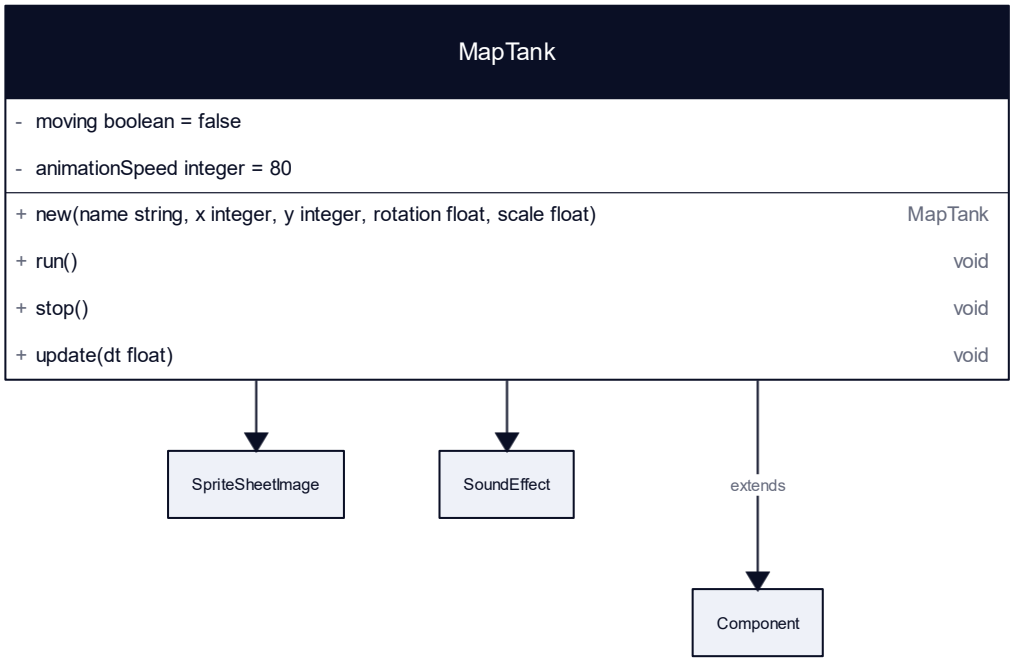
models/levelselect/LevelButton.lua

Le composant LevelButton est le bouton qui permet de sélectionner un niveau



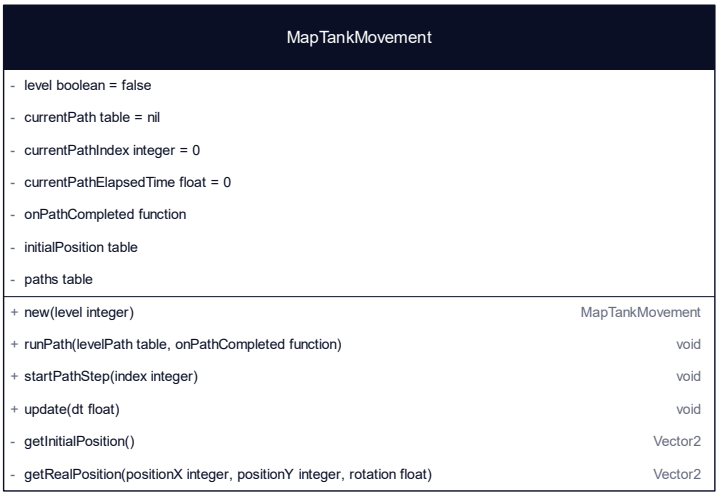
models/levelselect/MapTank.lua

Le composant MapTank est le composant qui affiche le tank sur la map



models/levelselect/MapTankMovement.lua

La classe MapTankMovement est la classe qui contient les paramètres de déplacement du tank sur la carte



Les organismes de la scène : Niveau du jeu

Les organismes de la scène Niveau du jeu sont les organismes qui composent la scène de Niveau du jeu.

J'ai séparé les organismes de niveau de jeu en 6 catégories par catégories fonctionnelles :

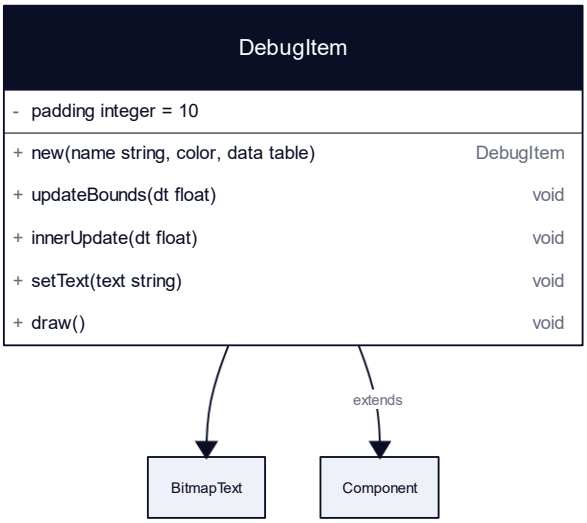
- Eléments de débogage
- Eléments de jeu
- Données du niveau de jeu
- Notifications en jeu
- Eléments liés à la pause
- Eléments de l'interface

Les organismes de la scène : Niveau du jeu- Débogage

Les organismes de la scène Niveau du jeu débogage sont l'ensemble des organismes qui permettent de débuguer la scène de Niveau du jeu.

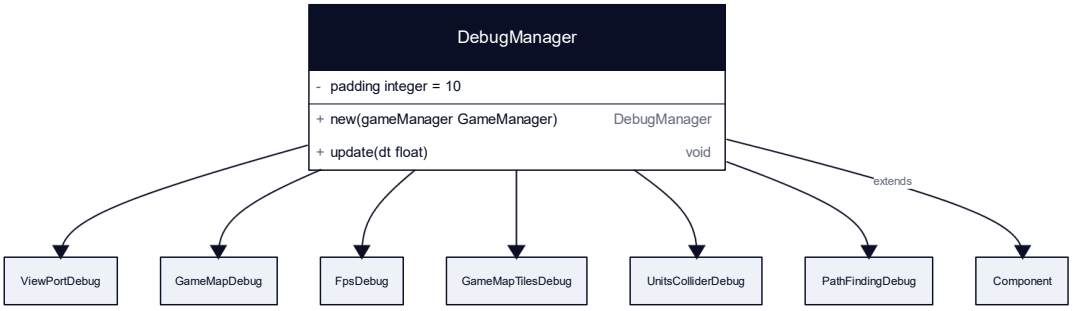
models/gameLevel/debug/DebugItem.lua

Le composant DebugItem est le composant de base de chaque composant de débogage de la scène de Niveau du jeu.



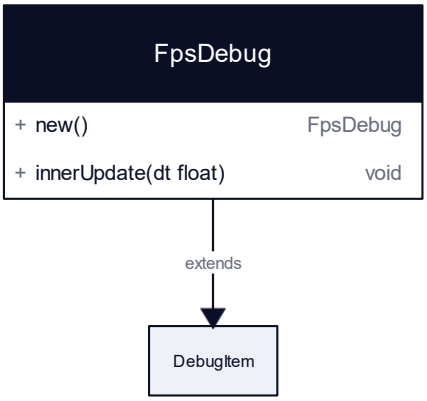
models/gameLevel/debug/DebugManager.lua

Le composant DebugManager gère la collection d'éléments de débogage de la scène de niveau de jeu.



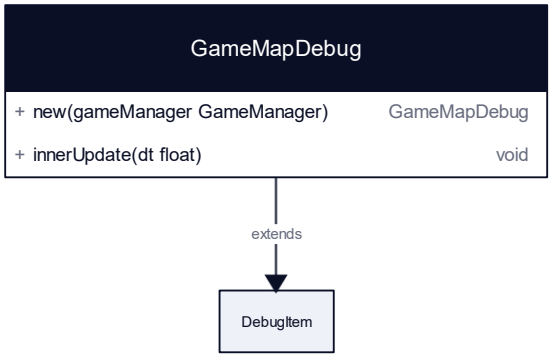
models/gameLevel/debug/FpsDebug.lua

Le composant FpsDebug permet d'afficher les fps du jeu en cours.



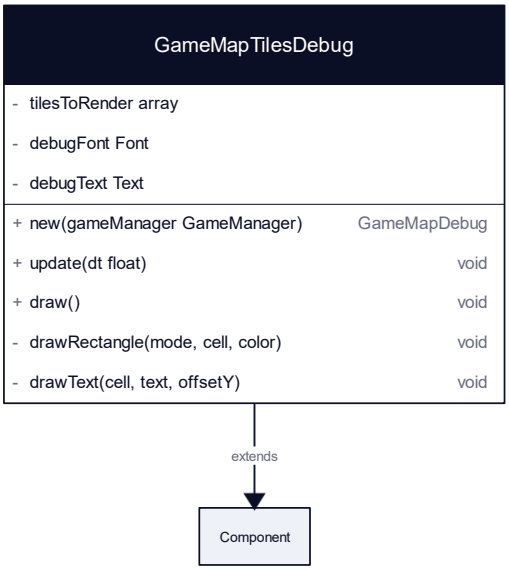
models/gameLevel/debug/GameMapDebug.lua

Le composant GameMapDebug permet d'afficher les informations de débogage de la carte (nombre de cellules affichées, etc.



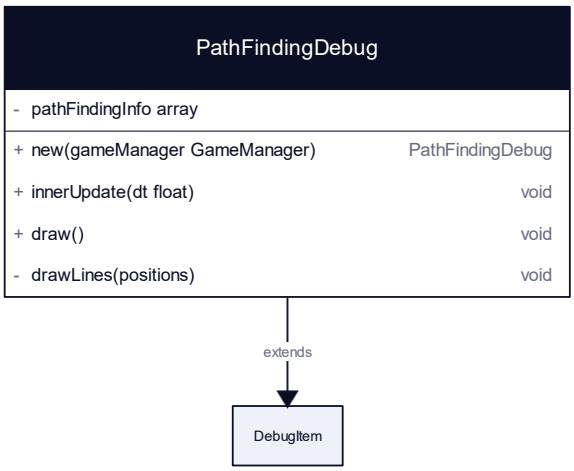
models/gameLevel/debug/GameMapTilesDebug.lua

Le composant GameMapTilesDebug permet d'afficher les informations de débogage de la carte (index des tuiles, etc.)



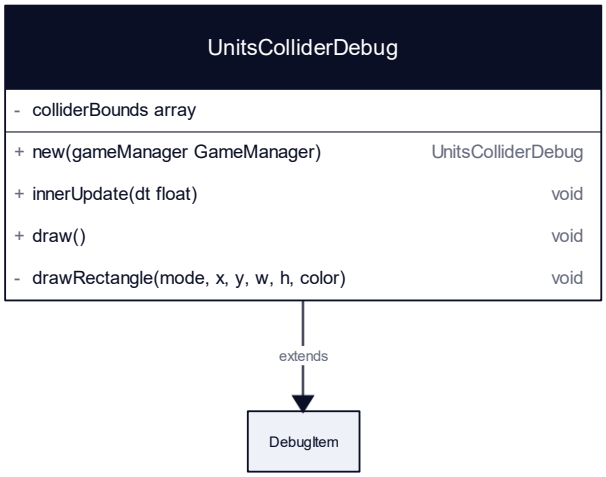
models/gameLevel/debug/PathFindingDebug.lua

Le composant PathFindingDebug permet d'afficher à l'écran les informations de débogage du pathfinding.



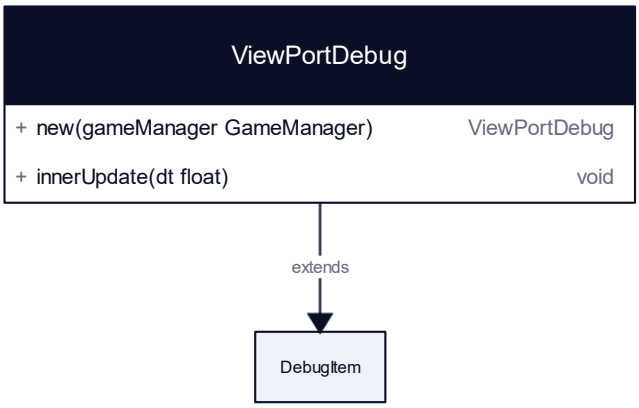
models/gameLevel/debug/UnitsColliderDebug.lua

Le composant UnitsColliderDebug permet d'afficher à l'écran les informations de collision des éléments du jeu.



models/gameLevel/debug/ViewPortDebug.lua

Le composant ViewPortDebug permet d'afficher à l'écran les informations sur le viewport de la scène (position de la caméra, etc.)



Les organismes de la scène : Niveau du jeu- Elément de jeu

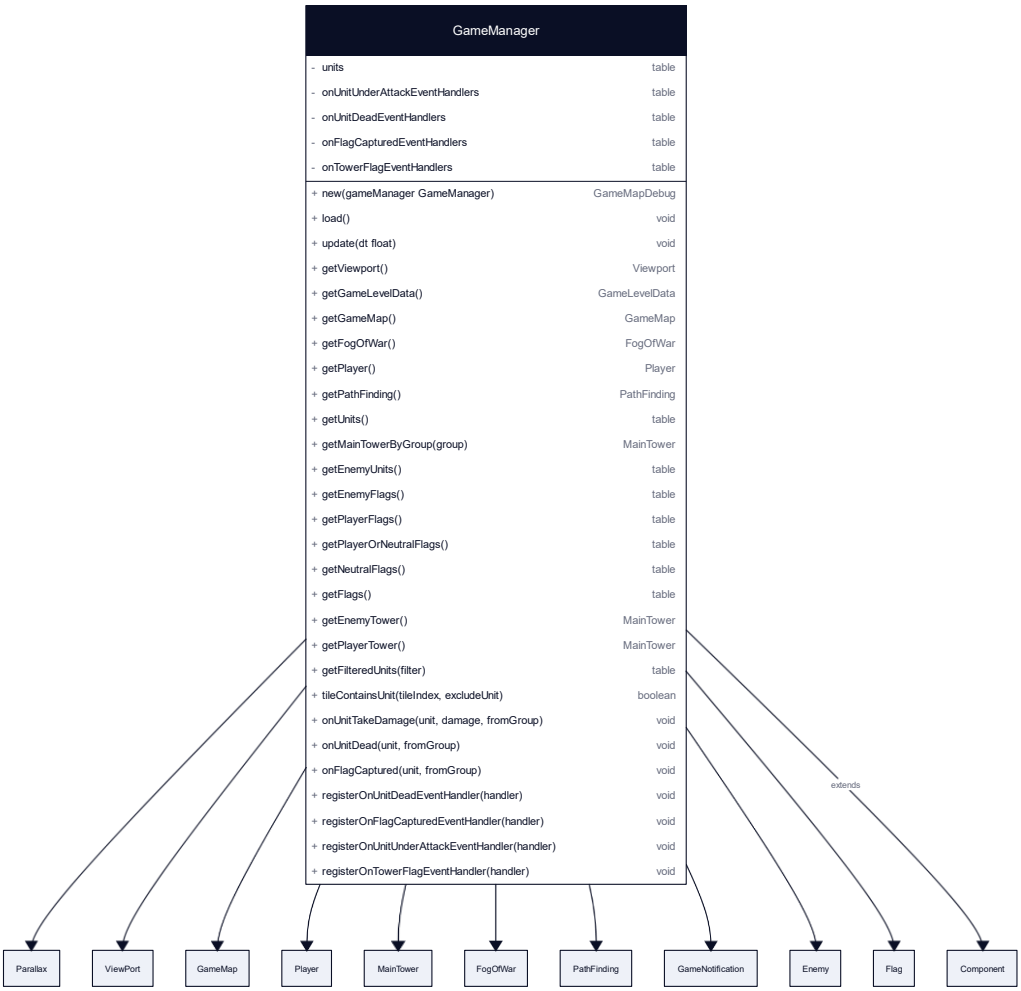
Les organismes de la scène Niveau du jeu élément de jeu sont l'ensemble des organismes qui donne vie au jeu.

Outre l'élément GameManager (élément principal du jeu), les organismes sont décomposés en 3 catégories :

- Gestion du viewport
- Gestion de la carte
- Gestion des entités

models/gameLevel/game/GameManager.lua

Le composant GameManager est l'élément central du jeu. Il gère l'ensemble des éléments du jeu.

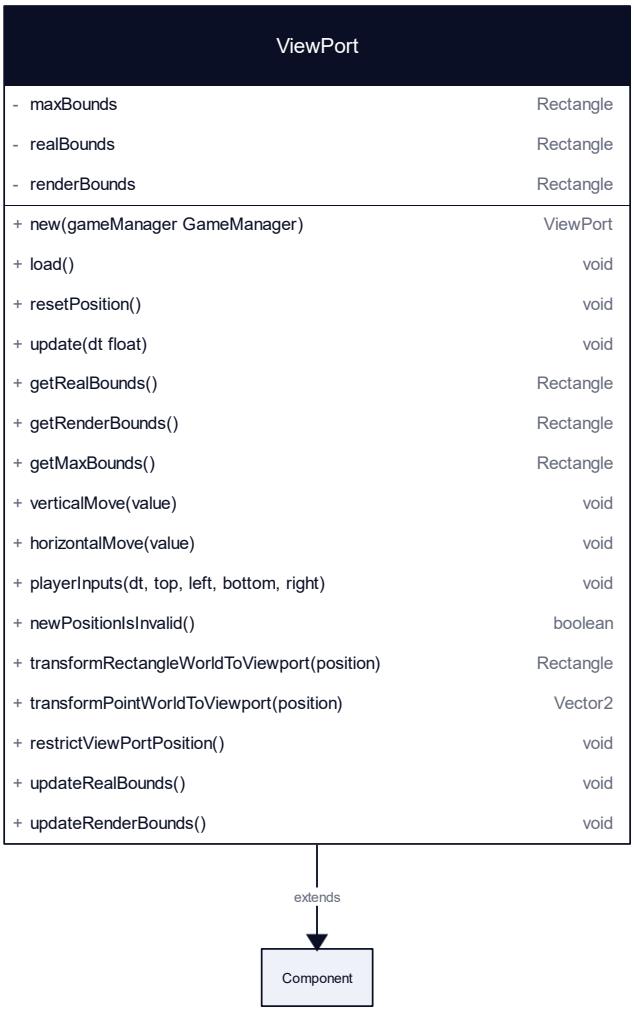


Les organismes de la scène : Niveau du jeu- Elément de jeu- ViewPort

Les organismes de la scène Niveau du jeu élément de jeu viewport sont les éléments qui gère l'affichage de la caméra sur la carte.

models/gameLevel/game/viewport/ViewPort.lua

Le composant ViewPort est l'élément qui gère l'affichage de la caméra sur la carte.

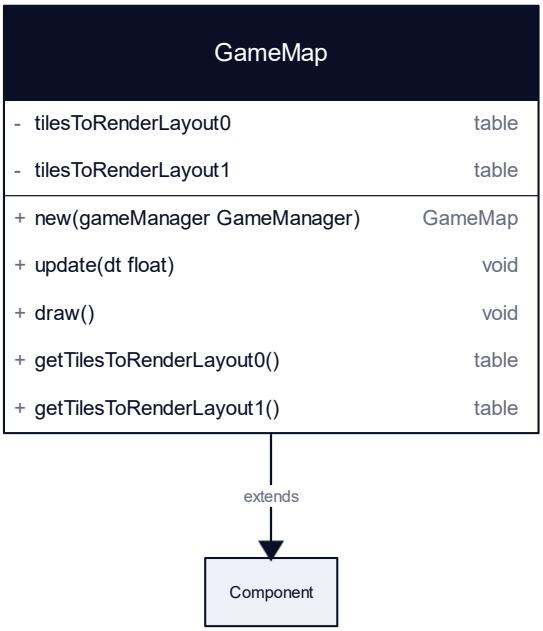


Les organismes de la scène : Niveau du jeu- Elément de jeu- Carte

Les organismes de la scène Niveau du jeu élément de jeu carte sont les éléments qui gère l'affichage de la carte.

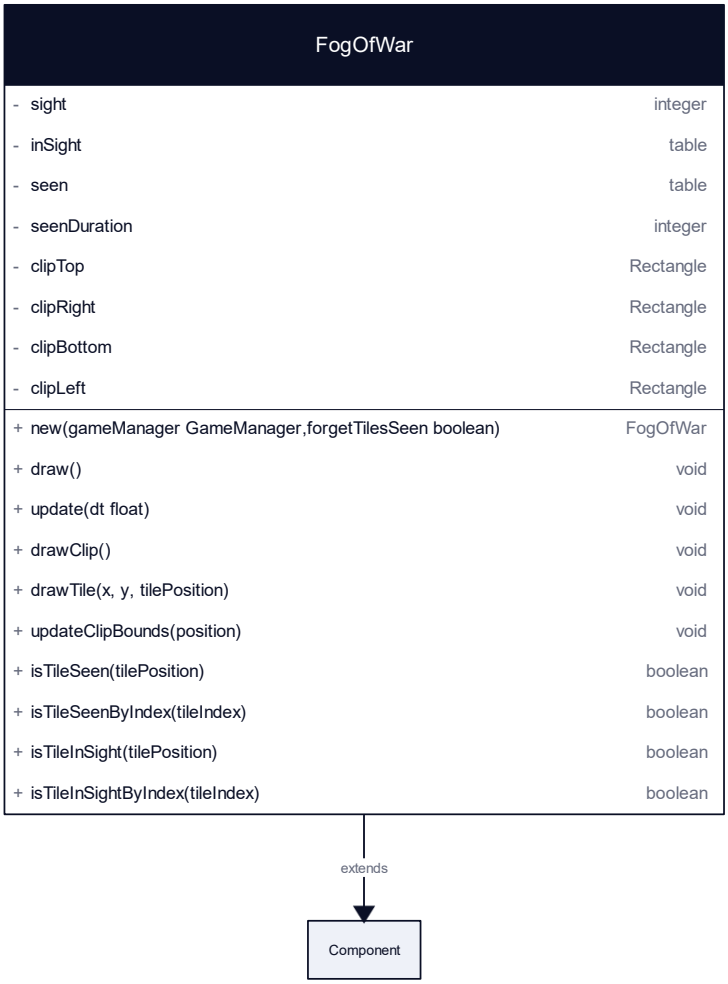
models/gameLevel/game/map/GameMap.lua

Le composant GameMap est l'élément qui affiche la carte.



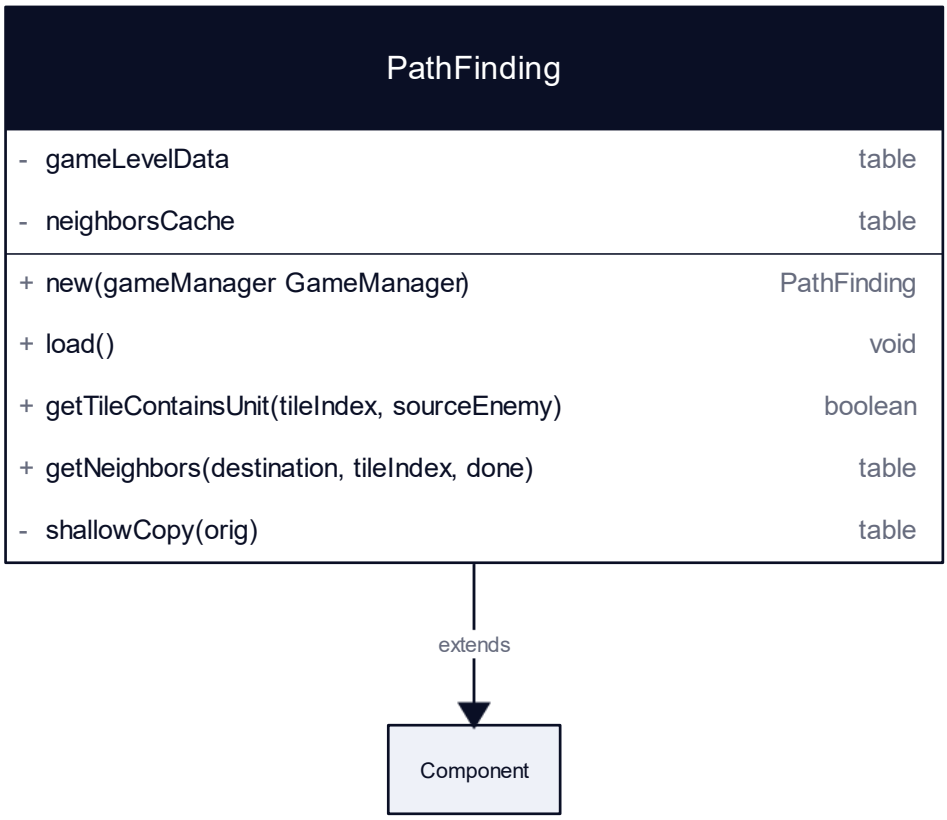
models/gameLevel/game/map/FogOfWar.lua

Le composant FogOfWar est l'élément qui affiche le brouillard de guerre.



models/gameLevel/game/map/PathFinding.lua

Le composant PathFinding est l'élément qui permet de calculer le déplacement des unités.



Les organismes de la scène : Niveau du jeu- Elément de jeu- Entités

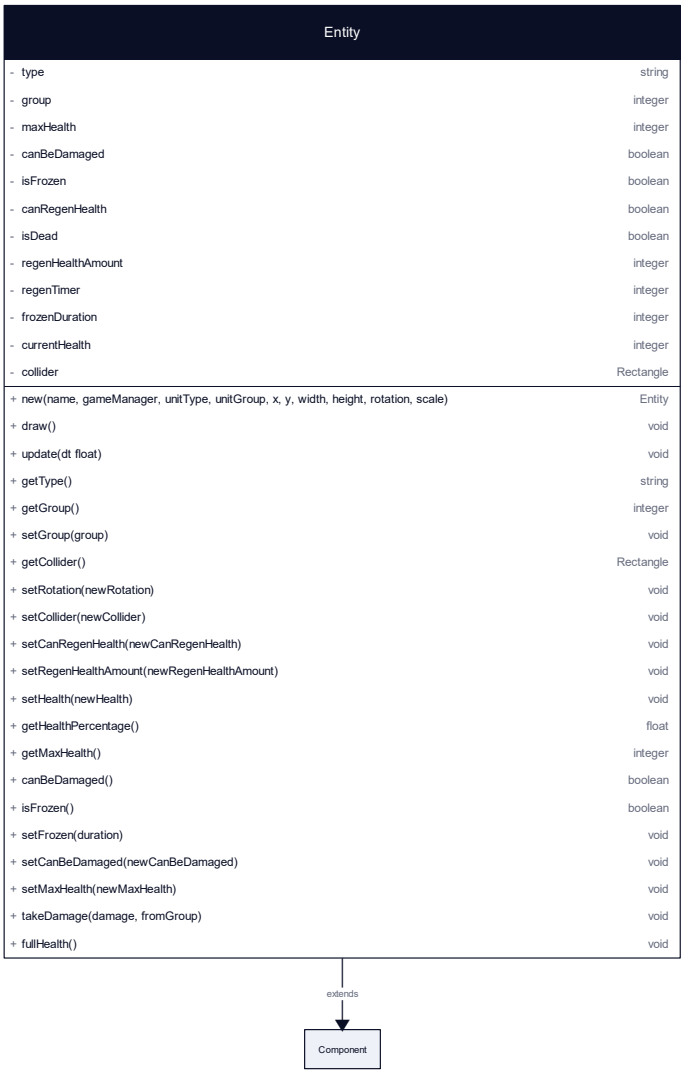
Les organismes de la scène Niveau du jeu élément de jeu entités sont les éléments qui gère les entités du jeu.

J'ai placé les éléments spécifiques à l'attitude des ennemis dans ce document :

- Attitude des ennemis

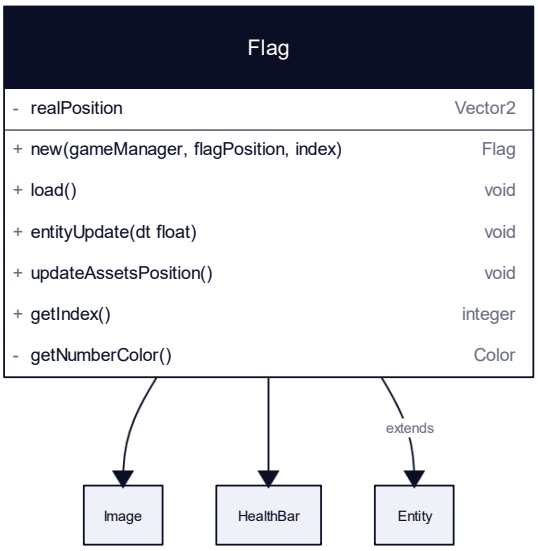
models/gameLevel/entities/Entity.lua

Le composant Entity est l'élément de base des entités du jeu.



models/gameLevel/entities/Flag.lua

Le composant Flag est l'élément qui représente les tours à capturer du jeu.

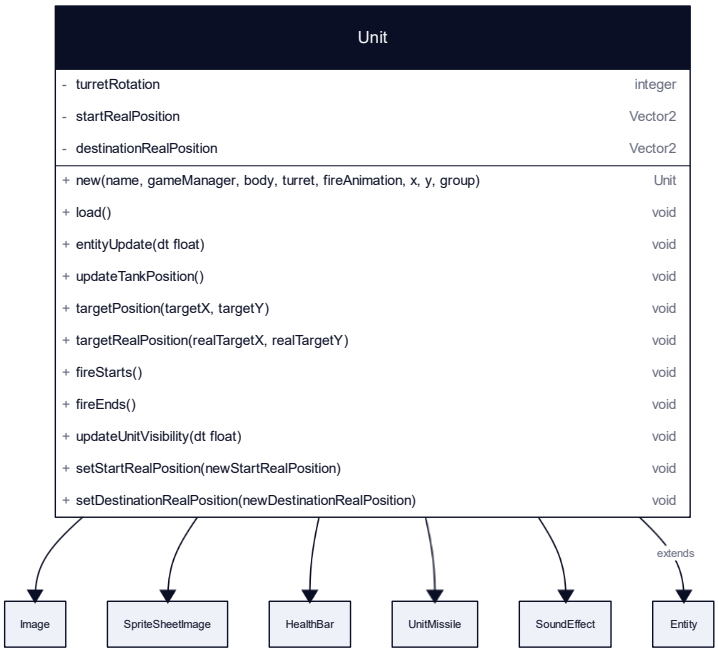


models/gameLevel/entities/MainTower.lua

Le composant MainTower est l'élément qui représente les tours principales des joueurs.

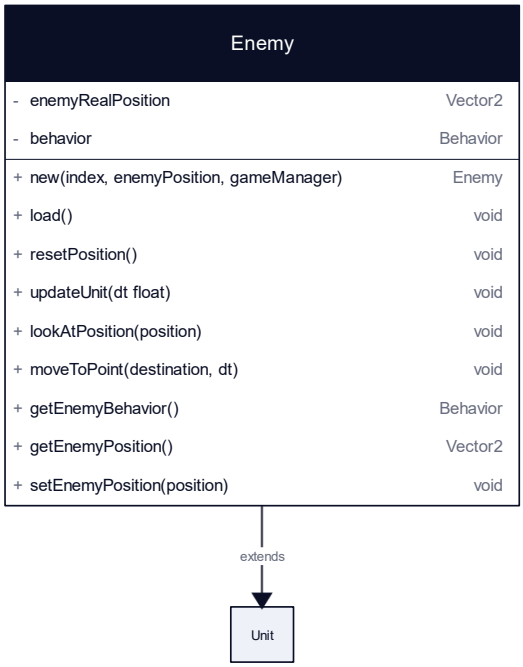
models/gameLevel/entities/Unit.lua

Le composant Unit est l'élément qui représente les unités (tank) du joueur ou des ennemis.



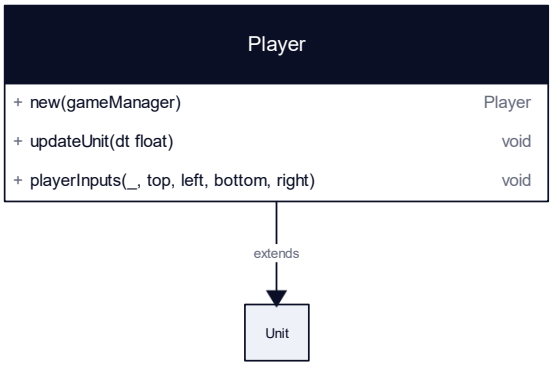
models/gameLevel/entities/Enemy.lua

Le composant Enemy est l'élément qui représente les unités (tank) de l'ennemi.



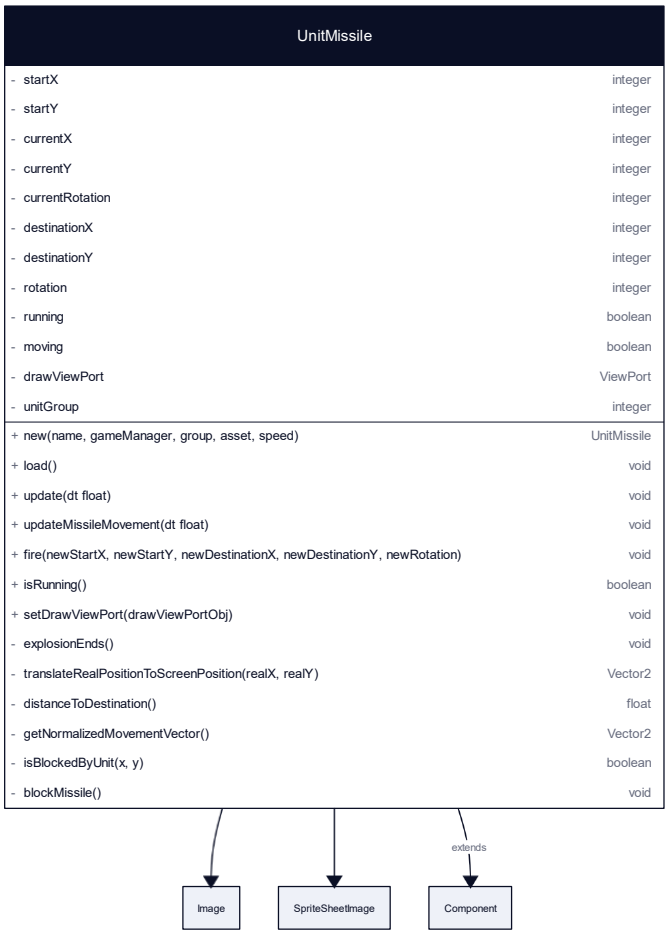
models/gameLevel/entities/Player.lua

Le composant Enemy est l'élément qui représente les unités (tank) du joueur.



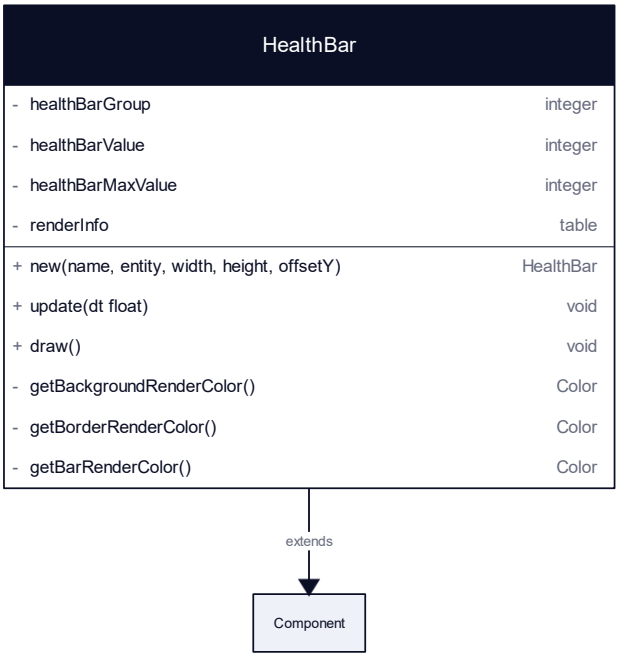
models/gameLevel/entities/UnitMissible.lua

Le composant UnitMissible est l'élément qui représente les missiles lancés par les entités.



models/gameLevel/entities/HealthBar.lua

Le composant HealthBar est l'élément qui représente les barres de vie des entités.



Les organismes de la scène : Niveau du jeu- Elément de jeu- Entités- Comportement

Eléments spécifiques à l'attitude des ennemis dans le jeu.

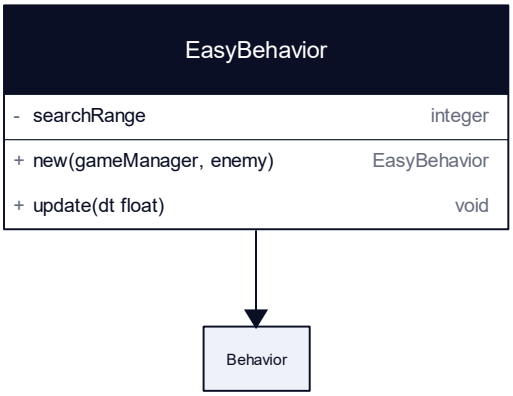
models/gameLevel/entities/behaviors/Behavior.lua

Le composant Behavior est l'élément de base pour le comportement des entités.

Behavior	
- currentOrder	Order
- seenTiles	table
- ignoredTiles	table
+ new(gameManager, enemy)	Behavior
+ update(dt float)	void
+ getFirstUnSeenTilePosition()	Vector2
+ setCurrentOrder(order)	void
+ resetCurrentOrder()	void
+ getCurrentOrder()	Order
- getSeenTiles()	table
- updateTilesSeen(sight)	void
# getClosestEmptyTileFromPoint(destination, position)	Vector2
# searchPlayerInRange(range)	Player
# searchFlagInRange(range)	Flag

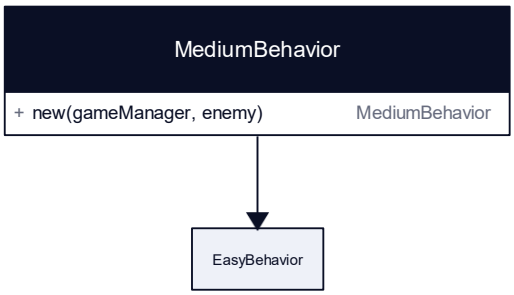
models/gameLevel/entities/behaviors/EasyBehavior.lua

Le composant EasyBehavior est l'élément de comportement des entités niveau facile.



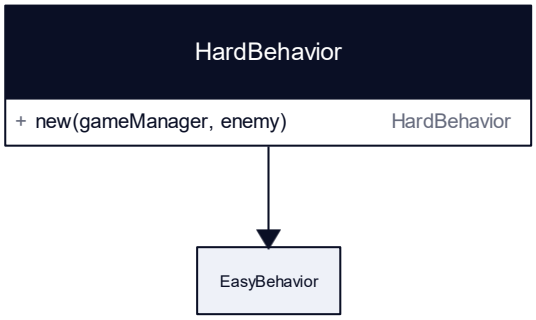
models/gameLevel/entities/behaviors/MediumBehavior.lua

Le composant MediumBehavior est l'élément de comportement des entités niveau moyen.



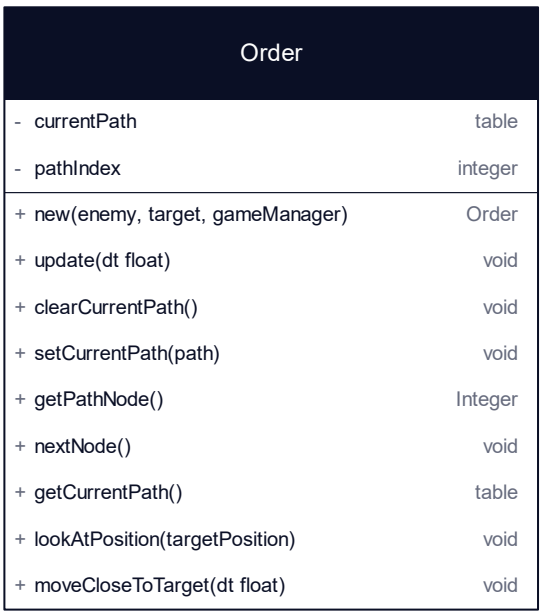
models/gameLevel/entities/behaviors/HardBehavior.lua

Le composant MediumBehavior est l'élément de comportement des entités niveau difficile.



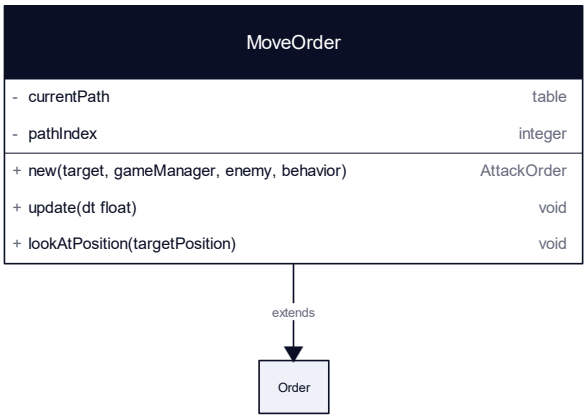
models/gameLevel/entities/behaviors/orders/Order.lua

Le composant Order est l'élément de base des ordres des entités.



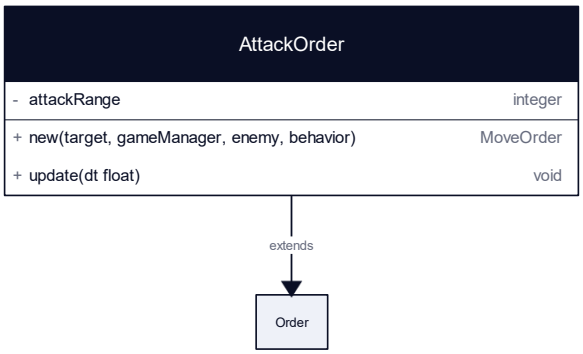
models/gameLevel/entities/behaviors/orders/MoveOrder.lua

Le composant MoveOrder est l'élément qui gère le déplacement d'une entités.



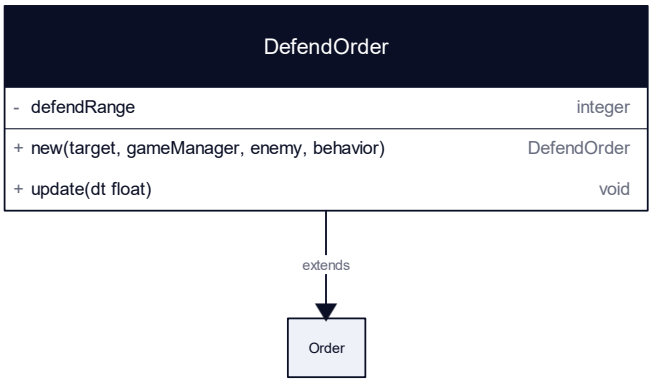
models/gameLevel/entities/behaviors/orders/AttackOrder.lua

Le composant AttackOrder est l'élément qui gère l'attaque d'une entités.



models/gameLevel/entities/behaviors/orders/DefendOrder.lua

Le composant DefendOrder est l'élément qui gère le passage en défense d'une entités.



Les organismes de la scène : Niveau du jeu- Données de jeu

Les organismes de la scène Niveau du jeu données de jeu sont l'ensemble des organismes qui permettent de stocker et gérer les données de la scène de Niveau du jeu.

models/gameLevel/levelData/GameLevelData.lua

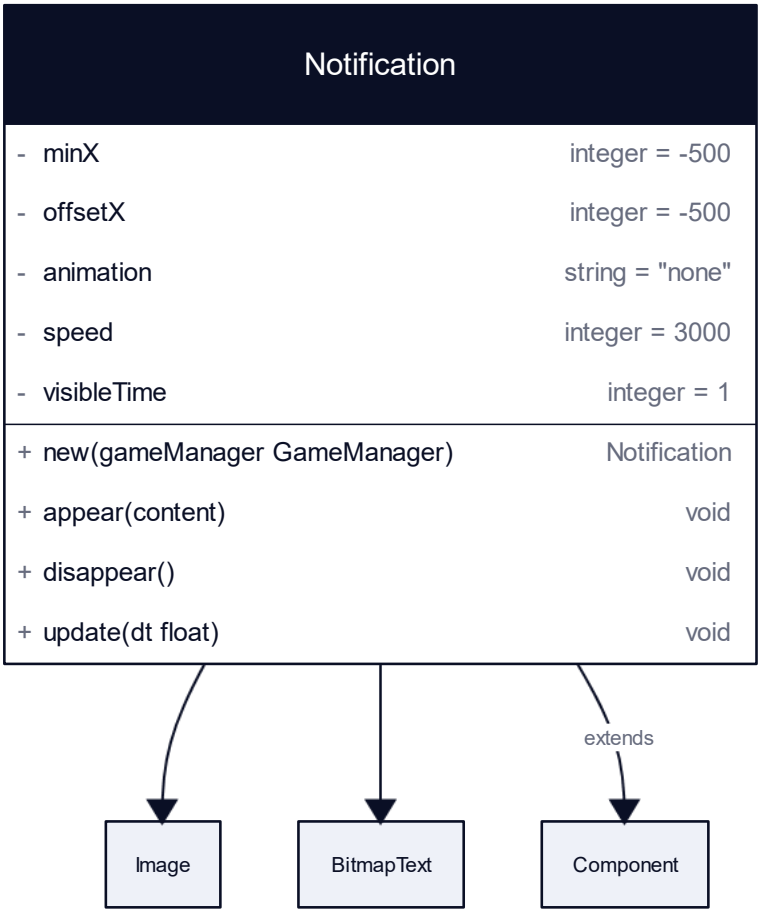
Le composant GameLevelData est le composant qui contient l'ensemble des données de la carte du jeu (textures des cellules, cellules bloqués etc.)

Les organismes de la scène : Niveau du jeu- Notifications

Les organismes de la scène Niveau du jeu notifications regroupent les organismes qui permettent d'afficher au joueur les notifications en jeu.

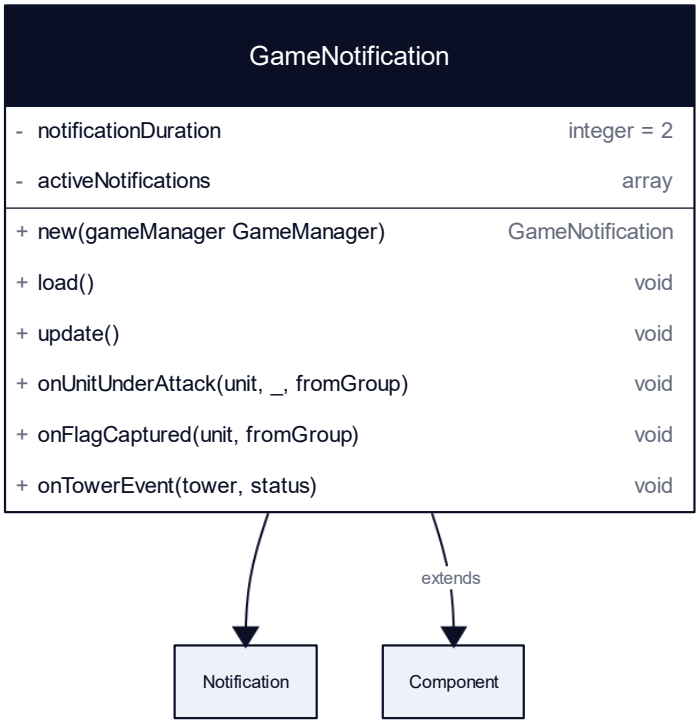
models/gameLevel/notifications/Notification.lua

Le composant Notification représente une notification en jeu.



models/gameLevel/notifications/GameNotification.lua

Le composant GameNotification est le composant qui affiche les notifications en jeu.

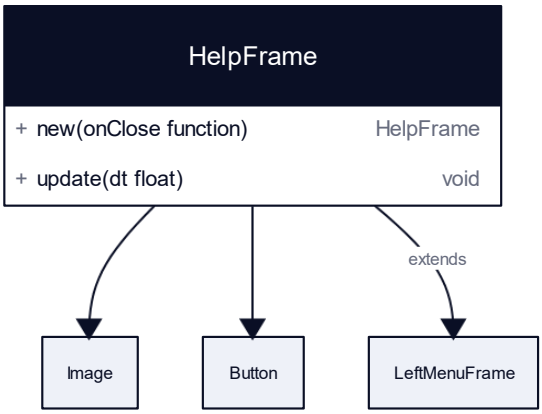


Les organismes de la scène : Niveau du jeu- Pause

Les organismes de la scène Niveau du jeu pause regroupent les organismes qui seront utilisés pour gérer la pause du jeu.

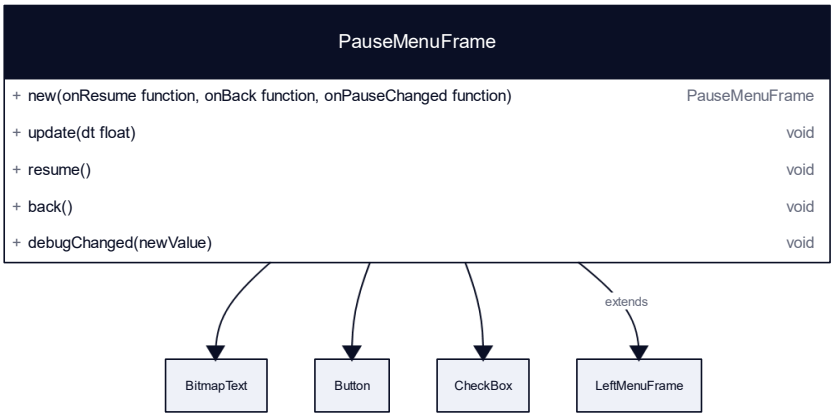
models/gameLevel/pause/HelpFrame.lua

Le composant HelpFrame affiche l'écran indiquant le but du jeu au démarrage d'un niveau.



models/gameLevel/pause/PauseMenuFrame.lua

Le composant PauseMenuFrame affiche le menu de pause du jeu.

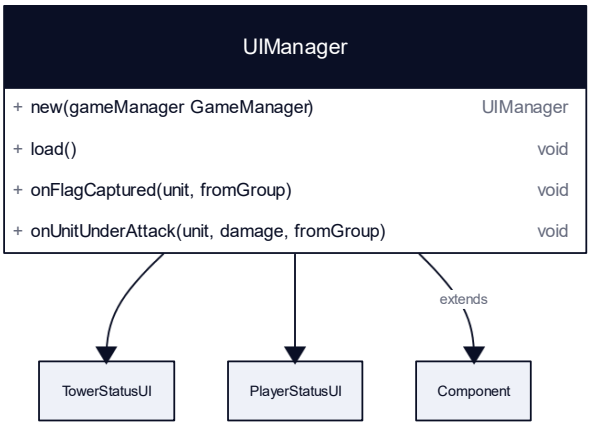


Les organismes de la scène : Niveau du jeu- Interface utilisateur

Les organismes de la scène Niveau du jeu interface utilisateur regroupent les organismes qui seront utilisés pour afficher l'interface utilisateur du jeu.

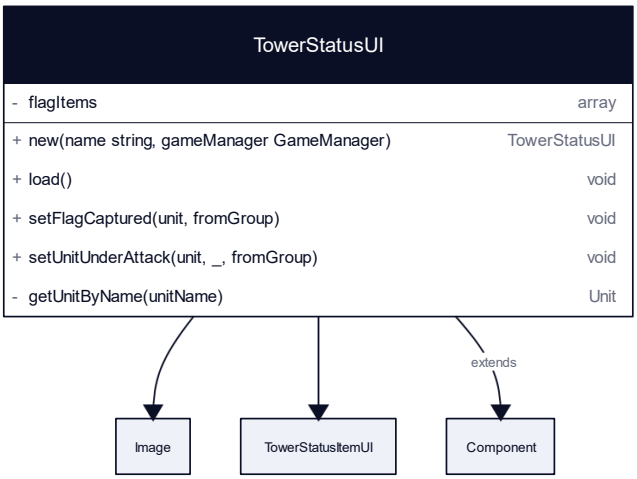
models/gameLevel/ui/UIManager.lua

Le composant UIManager affiche gère l'affichage des éléments de l'interface utilisateur en cours de jeu.



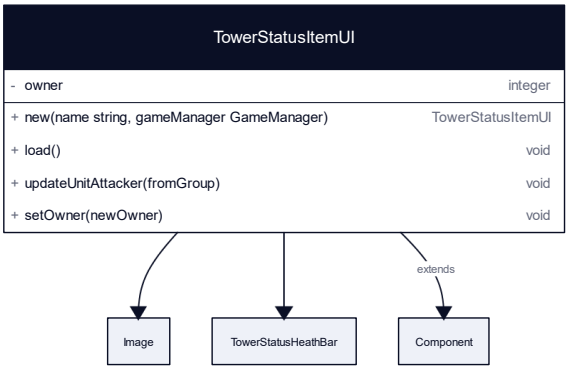
models/gameLevel/ui/TowerStatusUI.lua

Le composant TowerStatusUI affiche le statut des tours en haut de l'écran.



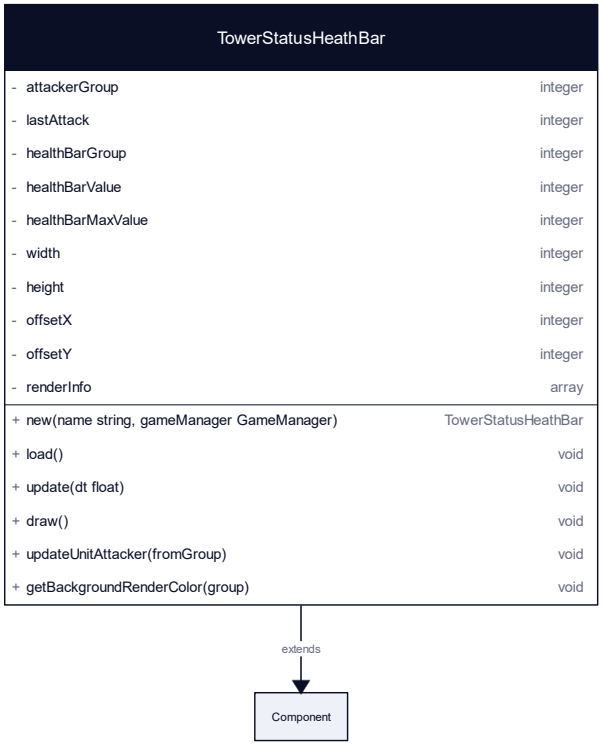
models/gameLevel/ui/TowerStatusItemUI.lua

Le composant TowerStatusItemUI affiche le statut d'une tour en haut de l'écran.



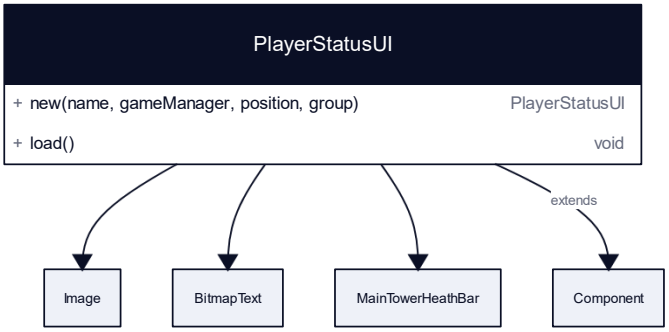
models/gameLevel/ui/TowerStatusHeathBar.lua

Le composant TowerStatusHeathBar affiche le barre de vie d'une tour en haut de l'écran.



models/gameLevel/ui/PlayerStatusUI.lua

Le composant PlayerStatusUI affiche les informations sur les tours principales en haut de l'écran (joueur ou ennemi).



models/gameLevel/ui/MainTowerHeathBar.lua

Le composant MainTowerHeathBar affiche la barre de vie des tours principales en haut de l'écran (joueur ou ennemi).

models/gameLevel/ui/EndGameUI.lua

Le composant EndGameUI affiche au joueur le message victoire ou défaite.