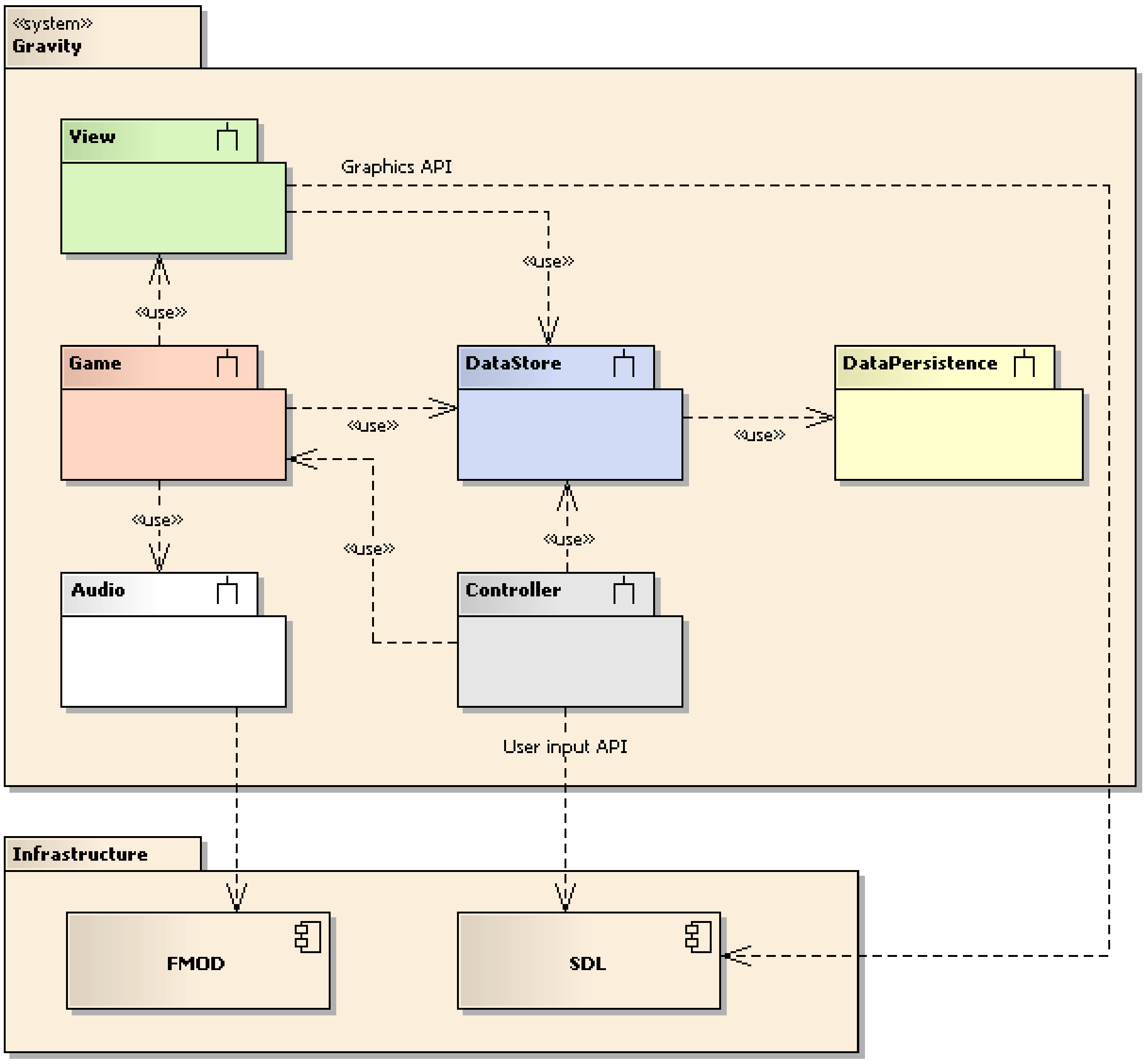
Gravity

Group 10

Daniel Walz
Jesper Ekhall
Lukas Kalinski
Fredrik Nordh
Alexander Nyberg

2.2

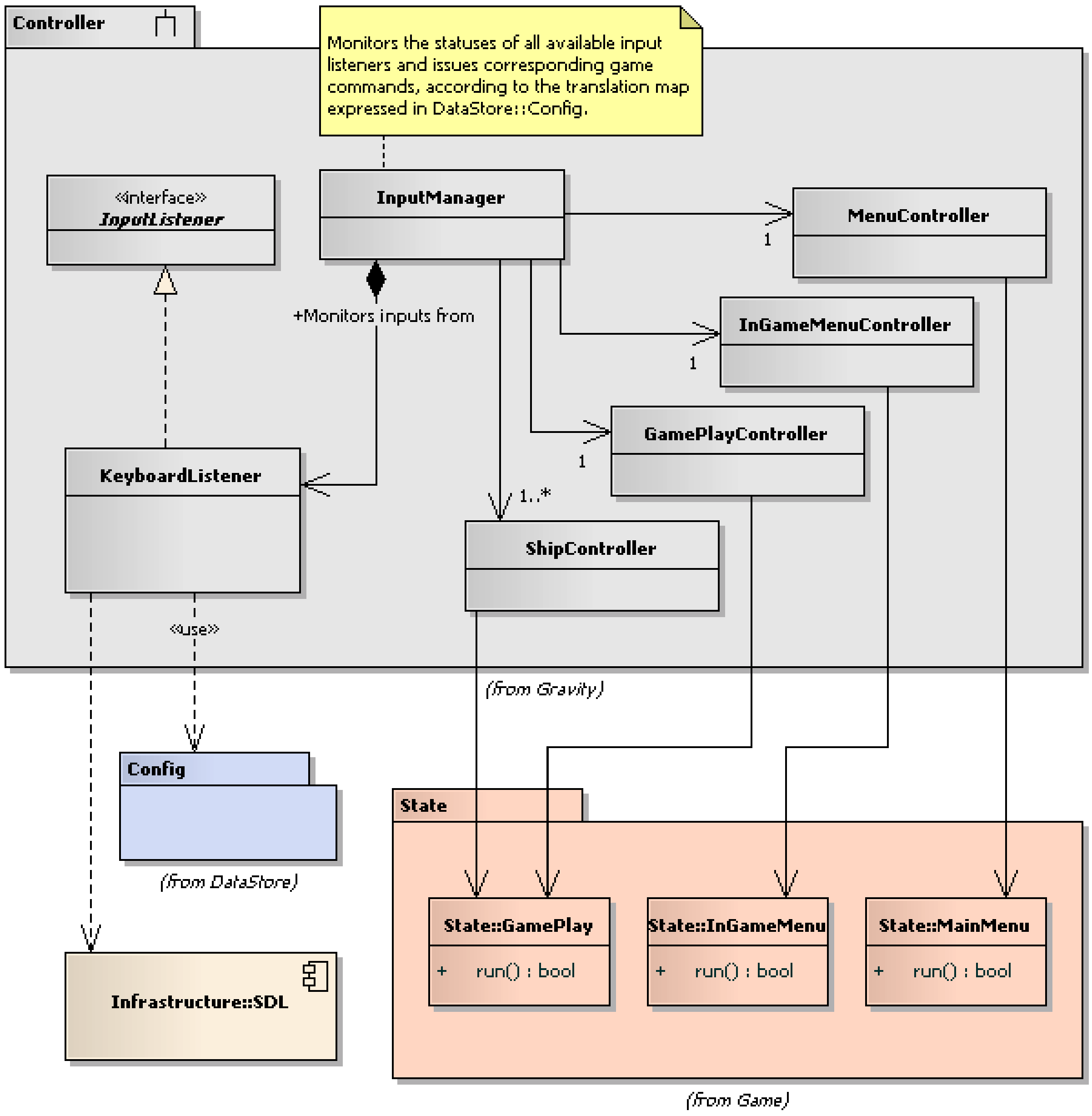


The system will consist of six modules. The Game module is the game core, this is where all game-related operations, such as affecting the game world using an internal physics package as well as switching between the different game states, will be done. The View and Audio modules will be used by the game module to trigger visual and auditory feedback on what's happening in the game world. Further, these modules will use third party tools to realize their purposes, i.e., the View module will use the Simple DirectMedia Layer (SDL) library to initialize and control the visual experience, and the Audio module will use the FMOD API to play sound effects. The Controller module will be responsible for interpreting user input into game commands, by using matching functionality in the SDL library. The DataStore module will manage shared data, such as for example a player's position in the game world, as well as the game world itself. The DataPersistence module will be used by the DataStore module to write persistent data into files.

2.3

Module: Controller

This module will contain everything that is related to user input and its consequences for the system flow.



Class: Input Manager

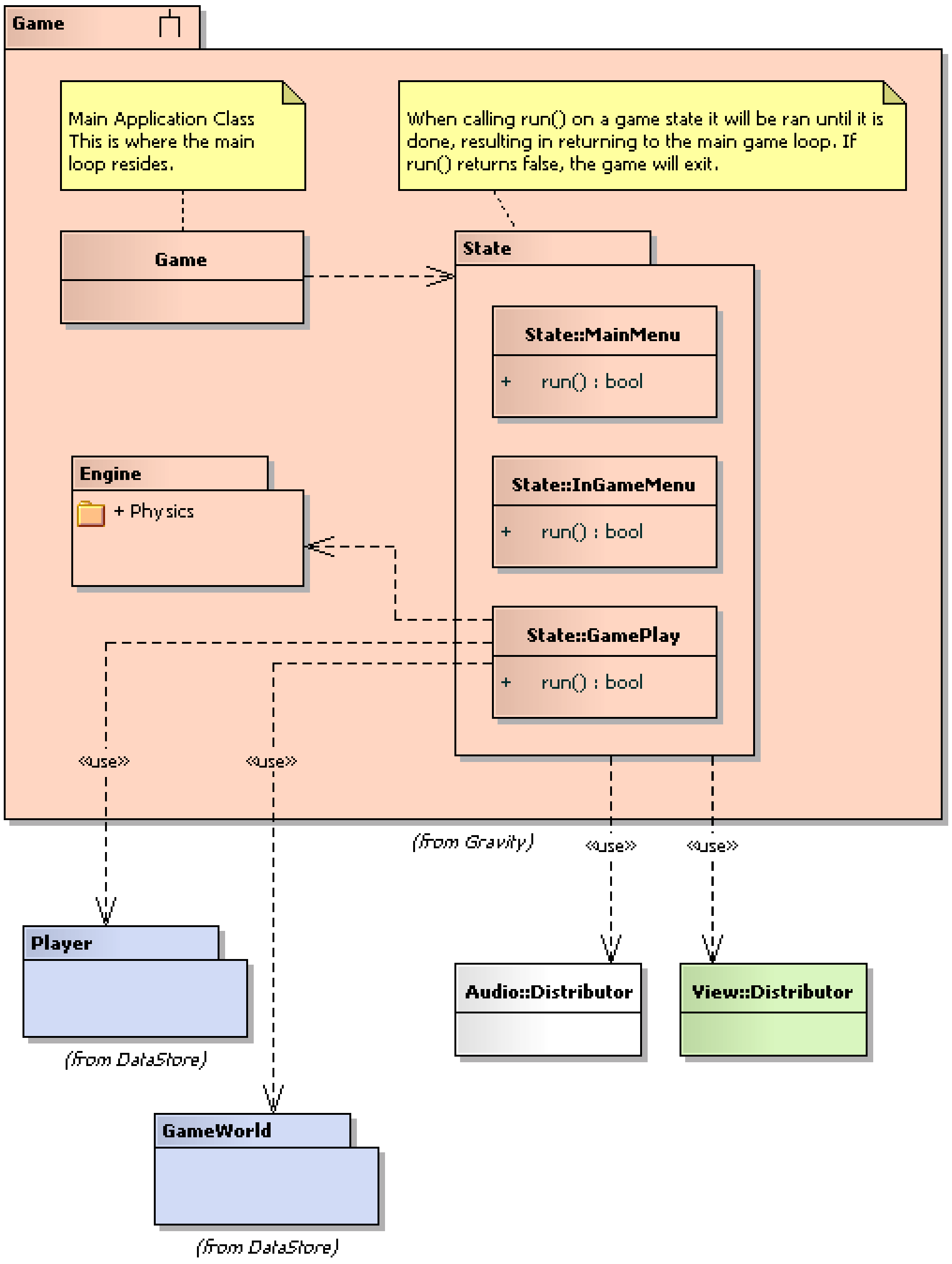
With future support for networking in mind, this is a crucial class within this module. It is able read input from anywhere (such as local keyboard or network) as it relies on an interface (InputListener) instead of hard-coded one-purpose input listening.

Class Group: Game Controllers

The game controller classes are responsible for issuing commands on different areas within the game, such as for example on a player's ship or the main menu. An example of a command is to go up in a menu or activate thrust on a ship.

Module: Game

This module will contain game logic, which includes the different states (menu, game play, etc.) and their logic, for example, the game play state will use the Physics Engine.



Class: Game

The starting point of the system, forwards main loop responsibility to different game states.

Package: State

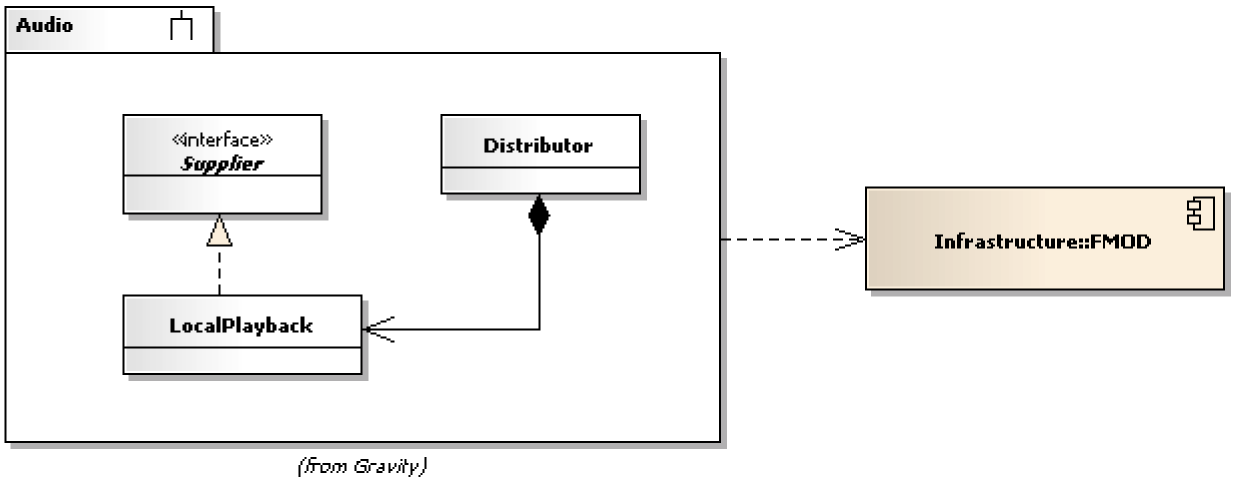
Contains representations of the states that the system may fall into. Each state is responsible for issuing a render of its own view. In addition, the game play state will also be responsible for issuing audio playback.

Package: Engine

Contains everything that is related to calculations within the game world and the menus. This for example includes the Physics Engine.

Module: Audio

This module will contain classes that are responsible for playing sounds.



Class: Distributor

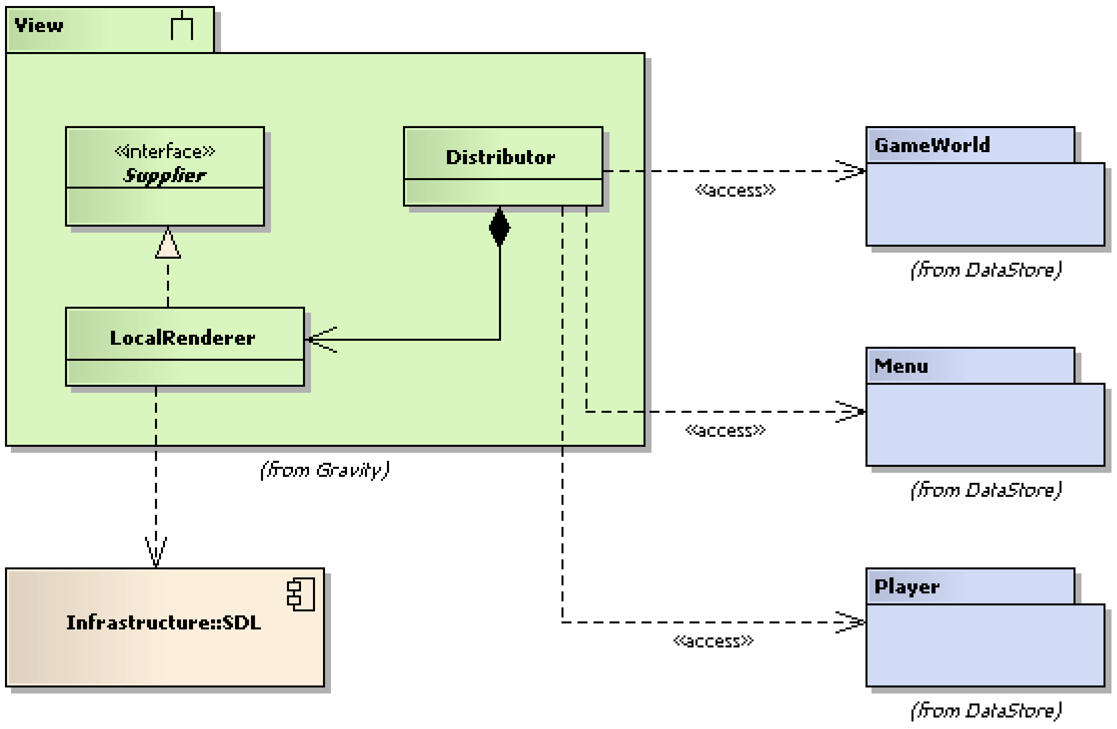
The Distributor is the heart of the audio module. It is responsible for “delivering” the sound to wherever it is to be delivered, relying on the Supplier interface. This property makes sound playback easy to integrate with a possible future networking environment.

Interface: Supplier

Defines the interface of an audio supplier, which basically is a class that is responsible for making one or more sounds heard somewhere.

Module: View

This module will contain classes that are responsible for rendering and displaying graphics.

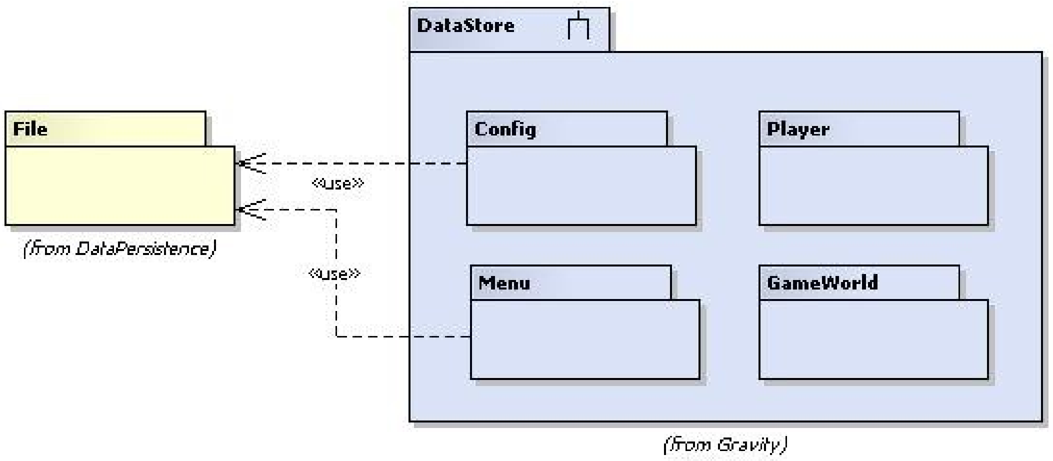


Class: Distributor

The Distributor is the heart of the view module. It is responsible for "delivering" a graphical view to wherever it is to be delivered, relying on the Supplier interface. This property makes view rendering easy to integrate with a possible future networking environment, for example by having a class for sending view data implement the Supplier Interface.

Module: Data Store

This module will contain shared data.



Package: Config

Will contain classes responsible for reading and writing configuration files.

Package: Menu

Will contain classes responsible for holding data about the game menus.

Package: Player

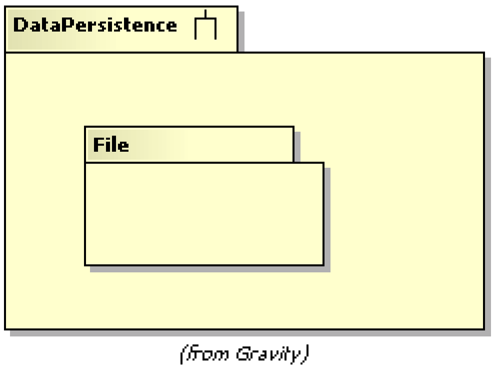
Will contain classes responsible for holding data about each player.

Package: GameWorld

Will contain classes responsible for holding data about the game world environment (such as planets, asteroids, etc).

Module: Data Persistence

This module will contain classes that are responsible for loading and saving persistent data.



Class: File

Class responsible for file management (read, write, etc).