

Pier436
Architecture Specification v2.0
Common Infrastructure Team

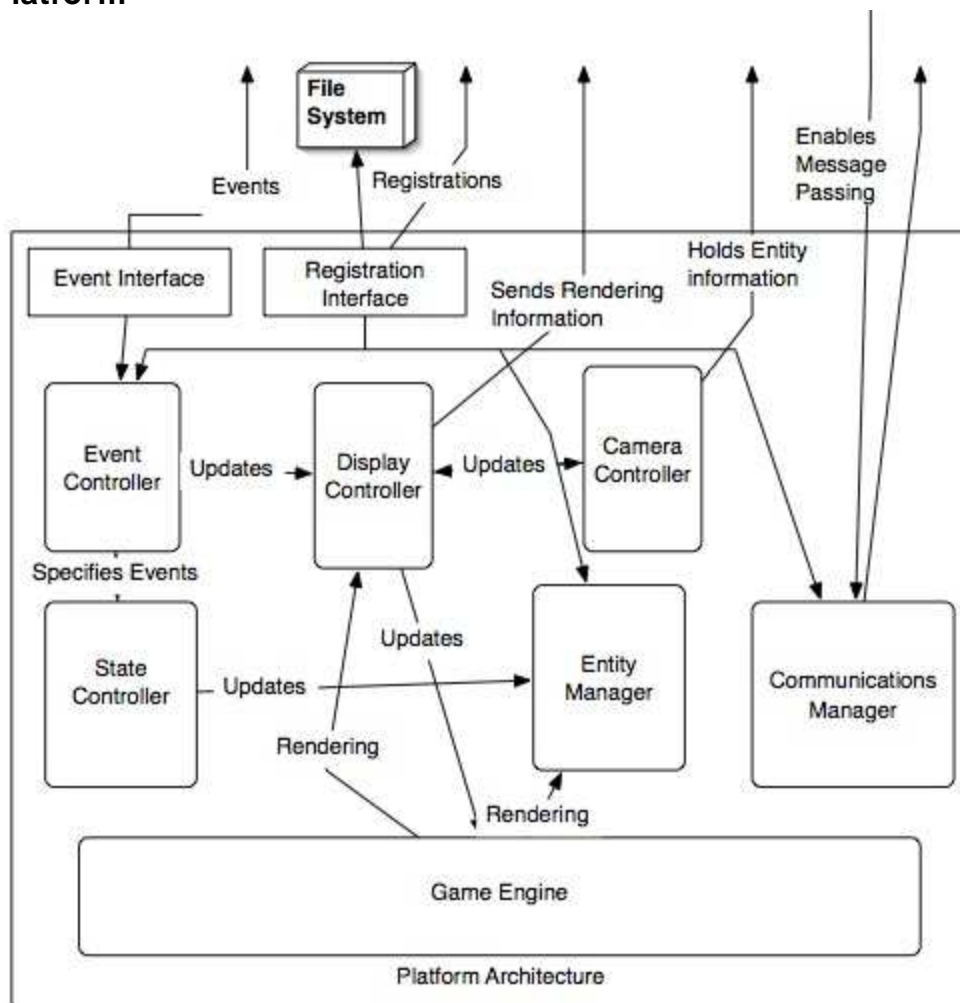
2 April, 2007

Matthew Bensley
Zachary Patterson
Troy Ruths
Jared Stein

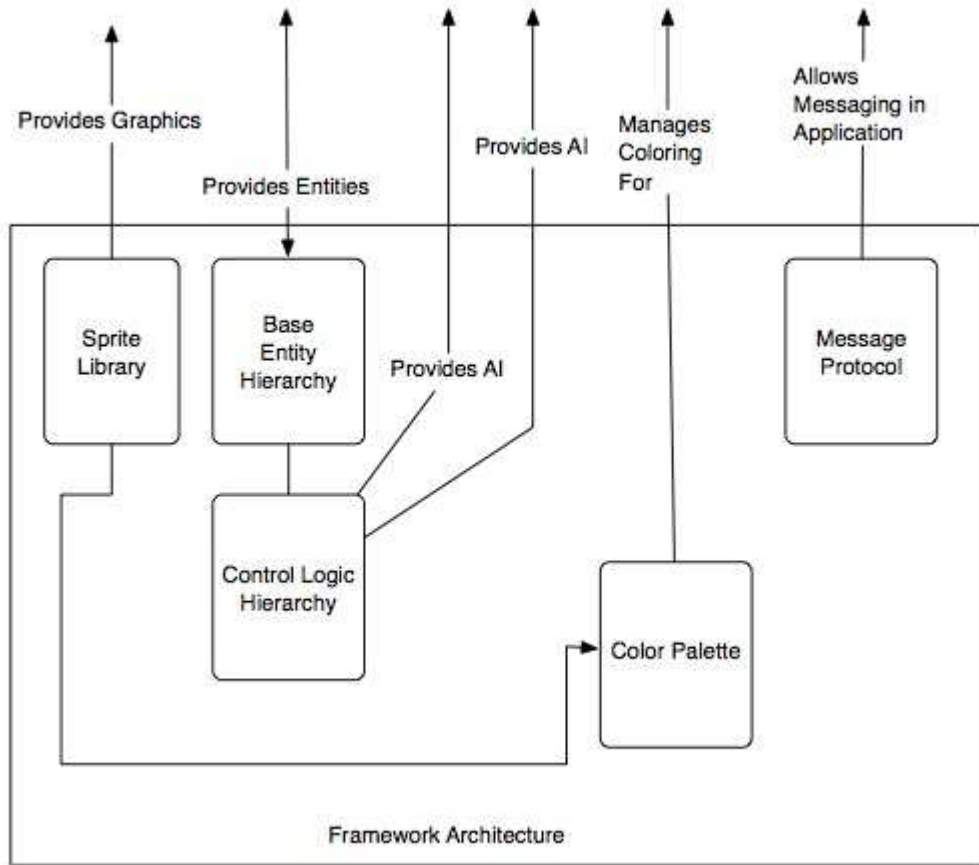
Pier436 is a distributed simulation environment which allows for creation and manipulation of entities across a networked which are rendered locally. The Platform deals with the dynamic activities while the Framework represents the static library type functions. Stepping forward into design, the Platform could shift into a server/client type approach; however the interface and functional abilities will remain intact. Ideally, the detached server should only be responsible for dispatching messages in order to maintain entity state across the application instances, so it will not affect other modules such as the camera controller. The Framework libraries will remain static as we approach a design.

Architecture Overview

1. Platform



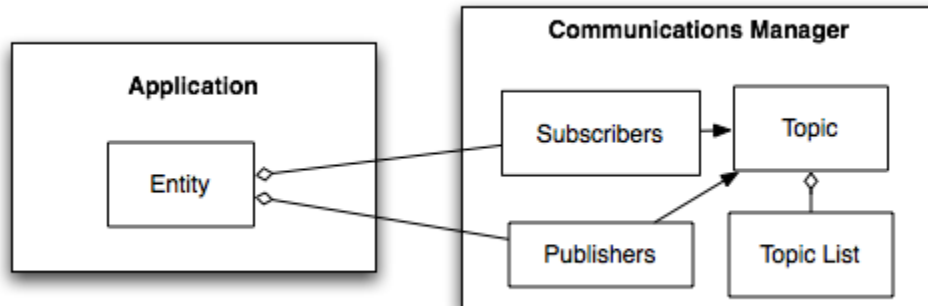
2. Framework



Architecture Components

1. Communications Manager

Assuming a publish/subscribe model, the communications manager can be decomposed in a very straightforward way. Assuming autonomy from the rest of the application, individual entities instantiated at the application level are added as publishers and subscribers, and message passing between components of different application instances works.



Responsibilities: Enable message passing between application instances. Built as external interface from platform to application layer.

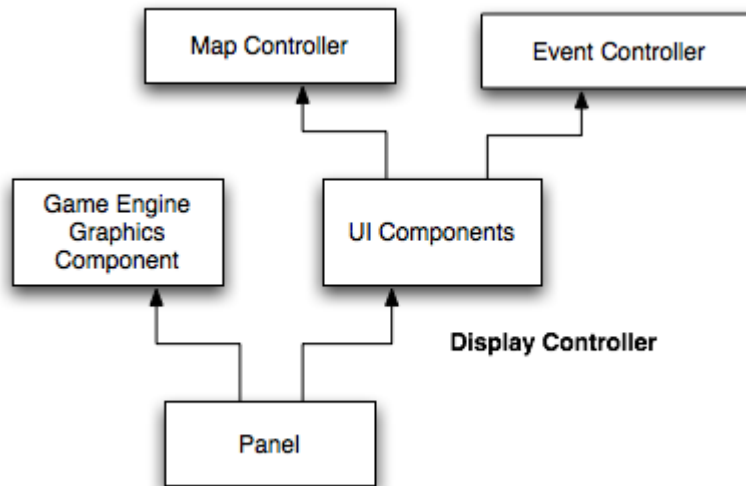
Dependencies: None.

Necessary for: Application message control center.

Requirements Numbers: 3.1.*

2. Display Controller

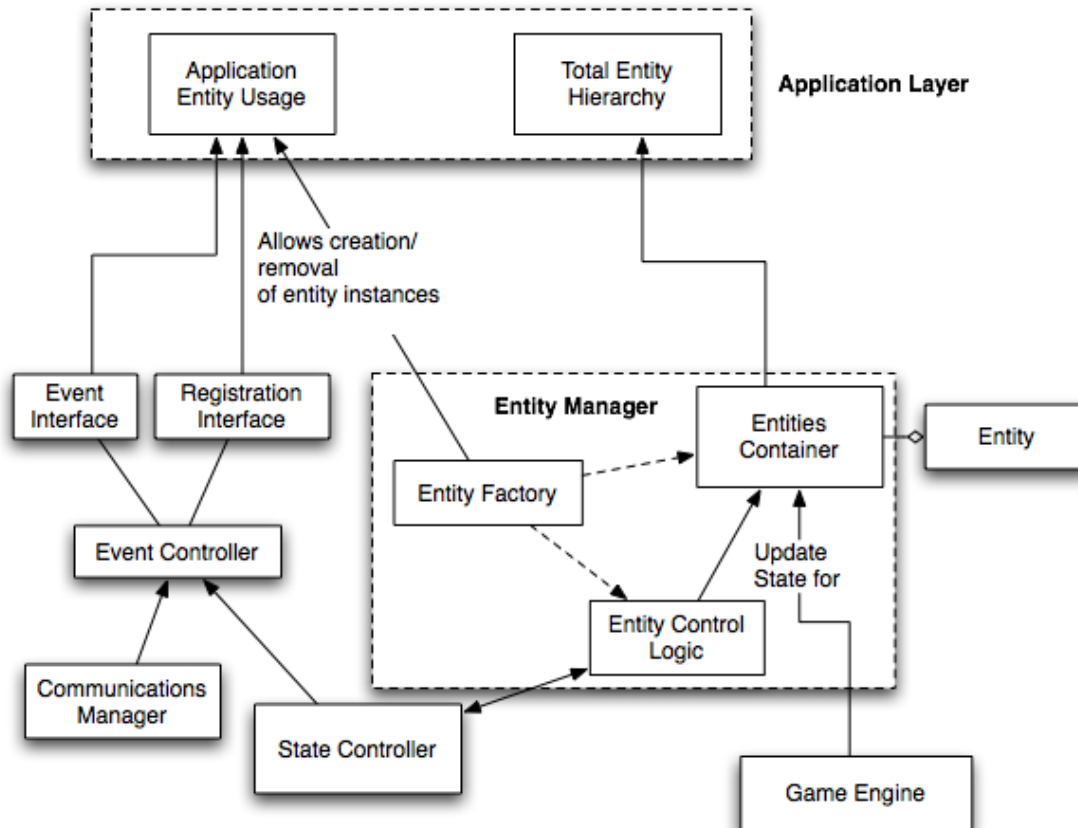
The display controller is composed of user interface components and the game engine graphics component, which are laid out on the panel, the general graphical user interface available to the client of the completed application. The UI Components will depend on some platform components as they will be used to create or affected by changes in game world state.



- Responsibilities: Accepts updates from events, the game engine, and the map controller within the platform and sends rendering information to the application layer. Accepts updates from the display and updates state in the map controller and game engine.
- Dependencies: Game engine, Map Controller, Event Controller
- Necessary for: Display, Map Controller
- Requirements Numbers: 1.2.*

3. Entity Manager

The entity manager is a data structure that holds information about each entity registered through the registration interface. The entity manager is necessary for the platform to effectively keep track of entities and notify them of events. The state controller and game engine will require frequent access to the entity manager in order to update entities and render them, respectively. While the entity manager holds references to each entity, the actual storage will occur in the application layer. The entity manager's objective is to allow other sub modules to be more effective, not to store entities.



Responsibilities: Holds entity information used in the application layer's entity storage structure.

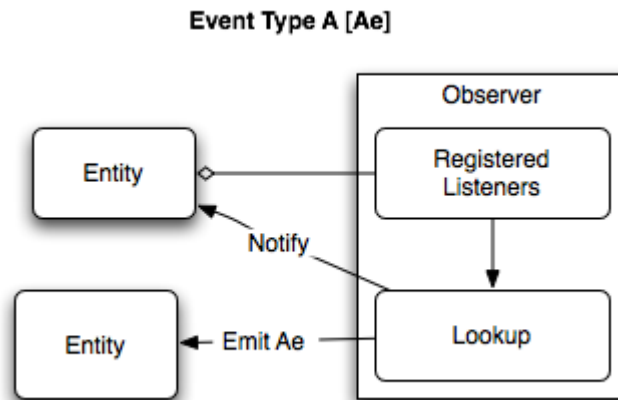
Dependencies: None.

Necessary for: Entity Storage, Game Engine

Requirements Numbers: 3.4.*

4. Event Controller

The event controller handles event notification and platform-side event handling. This includes both updating the display controller and passing events to the event interface module, which notifies entities of events. The registration interface also needs the event controller in order to register new entities for events. The event controller is abstracted away from the application layer using both the event interface and the registration interface, which now handle event notification and registration with the application.



Responsibilities: Receives messages from the application layer and updates the display and the world state. Notifies entities through the application's action manager.

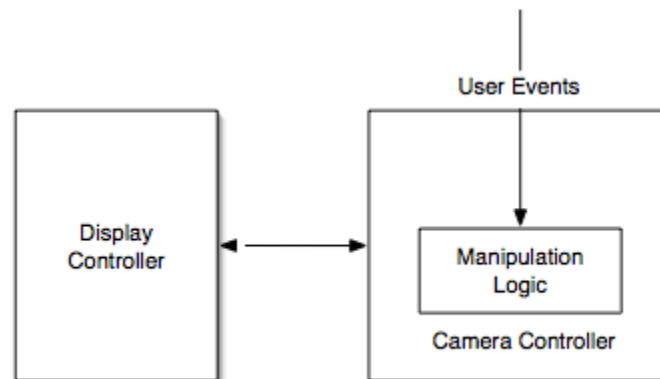
Dependencies: Application event creator, State Controller

Necessary for: Display Controller, State Controller

Requirements Numbers: 3.2.*

5. Camera Controller

Maintains the view of the map with respect to camera angle, zoom, and position and correspondingly updates the Display Controller.



Responsibilities: Updates the camera properties through the display controller.

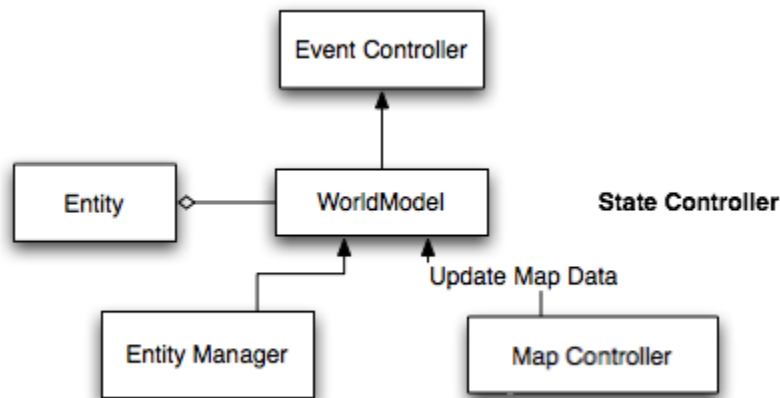
Dependencies: Display Controller

Necessary for: None

Requirements Numbers: 2.2.*, 4.3.*

6. State Controller

The state controller maintains time and allows for loading and resetting of the world model, which is just a collection of entities. It also updates the Entity Manager in response to the Event Controller.



Responsibilities: Holds world information and causes changes in entities. Also may cause events triggered by world state change.

Dependencies: Event Controller

Necessary for: Entity Manager, Event Controller

Requirements Numbers: 2.1.* , 3.3.*

7. Sprite Library

There is no decomposition since the Sprite Library consists of a set of files that contain the basic provided sprites.

Responsibilities: Basic 2D sprites for application teams.

Dependencies: None

Necessary for: Extended Entity Hierarchy and Display rendering

Requirements Numbers: 4.4

8. Base Entity Hierarchy

The complete entity hierarchy is yet to be determined, but will definitely provide for the entities listed in 4.1.* of the SRS.

Responsibilities: Base infrastructure for entity classes for application. Extendable by application layer.

Dependencies: None

Necessary for: Extended Entity Hierarchy

Requirements Numbers: 4.1.*

9. Control Logic Hierarchy

The complete control logic hierarchy is yet to be determined, but will

definitely provide for the control intelligences listed in 4.2.* of the SRS.

Responsibilities: Autonomous entity control and event reactions
Dependencies: None
Necessary for: Extended Entity Hierarchy, Event Action Manager
Requirements Numbers: 4.2.*

10. Color Palette

There is no decomposition for the Color Palette since it consists of the basic colors and tools for creating them through RGB inputs.

Responsibilities: Colors sprites on display.
Dependencies: Sprite Library
Necessary for: Display
Requirements Numbers: 4.6

11. Message Protocol

The Message Protocol is not decomposed since it is a format rather than a component.

Responsibilities: Provide standard message format for application message passing.
Dependencies: None
Necessary for: Application message control
Requirements Numbers: 4.5

12. Game Engine

Possible Game Engines: JMonkeyEngine, GTGE Game Engine (2D)
Responsibilities: Provides the graphical environment for viewing the application's world.
Dependencies: Entity Manager, Display Controller
Necessary for: None
Requirements Numbers: 1.1.*

Rationale

Platform

The platform provides the collaborative, graphical environment on top of which applications will run. Applications will make use of three public interfaces the Event Interface, the Registration Interface, and the Communications Manager. Events will be logged by the Communications Manager and notify the Event Controller which will update the Display Controller and the State Controller. Registration updates the Event Controller, updates the Entity Manager, and is logged by the Communications Manager. Internally, there are several more components handling tasks. The Camera Controller updates and is updated by

the Display Controller. The State Controller updates the Entity Manager. The Game Engine renders the Entities in the Entity Manager on the Display Controller.
Framework

The framework will be used by applications to provide common services. The Sprite Library uses the Color Palette to provide graphics. The Base Entity Hierarchy provides entities using the Control Logic Hierarchy. The Color Palette provides colors. The Message Protocol allows for messaging by the applications.