

# Pier436

Architecture Design v1.0  
Common Infrastructure Team

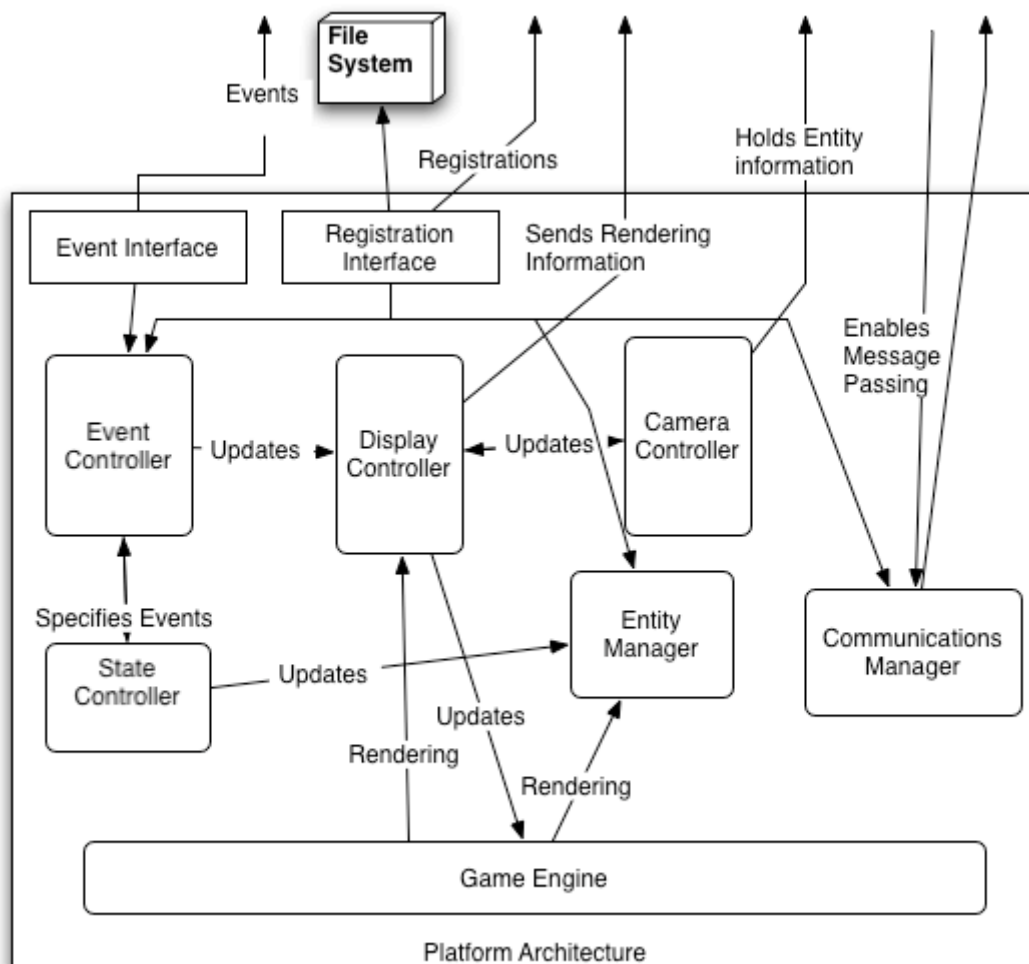
2 April 2007

*Matthew Bensley*  
*Zachary Patterson*  
*Troy Ruths*  
*Jared Stein*

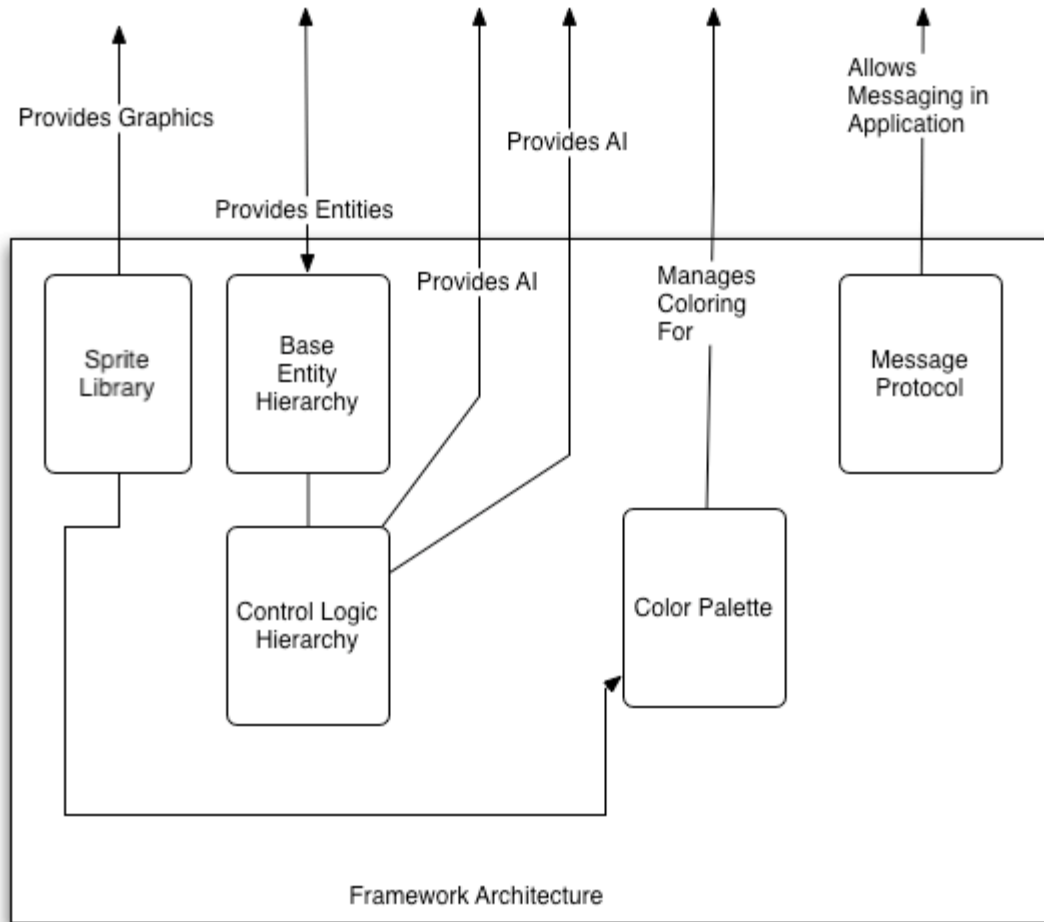
Pier436 is a distributed simulation environment which allows for creation and manipulation of entities across a networked which are rendered locally. The Platform deals with the dynamic activities while the Framework represents the static library type functions. Stepping forward into design, the Platform 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 effect 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

The communications manager is now for the most part satisfied by the JGN component. This will enable synchronization over multiple instances. In addition, we can create messages for our own types and send them across the network as required.

[www.captiveimagination.com](http://www.captiveimagination.com)

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

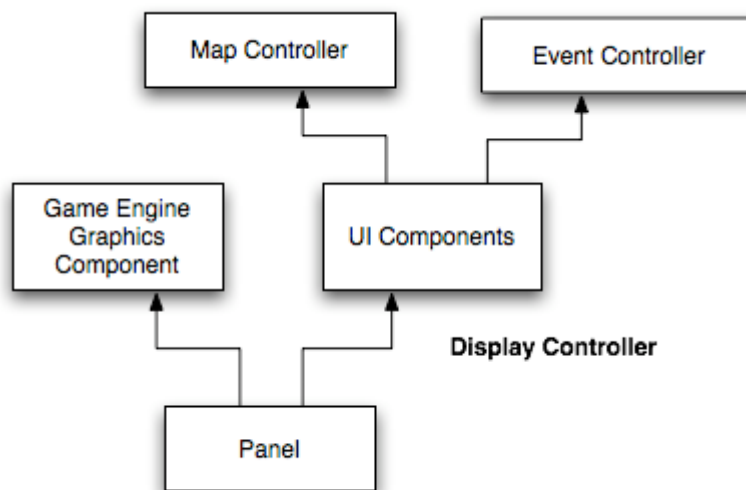
Dependencies: None.

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.

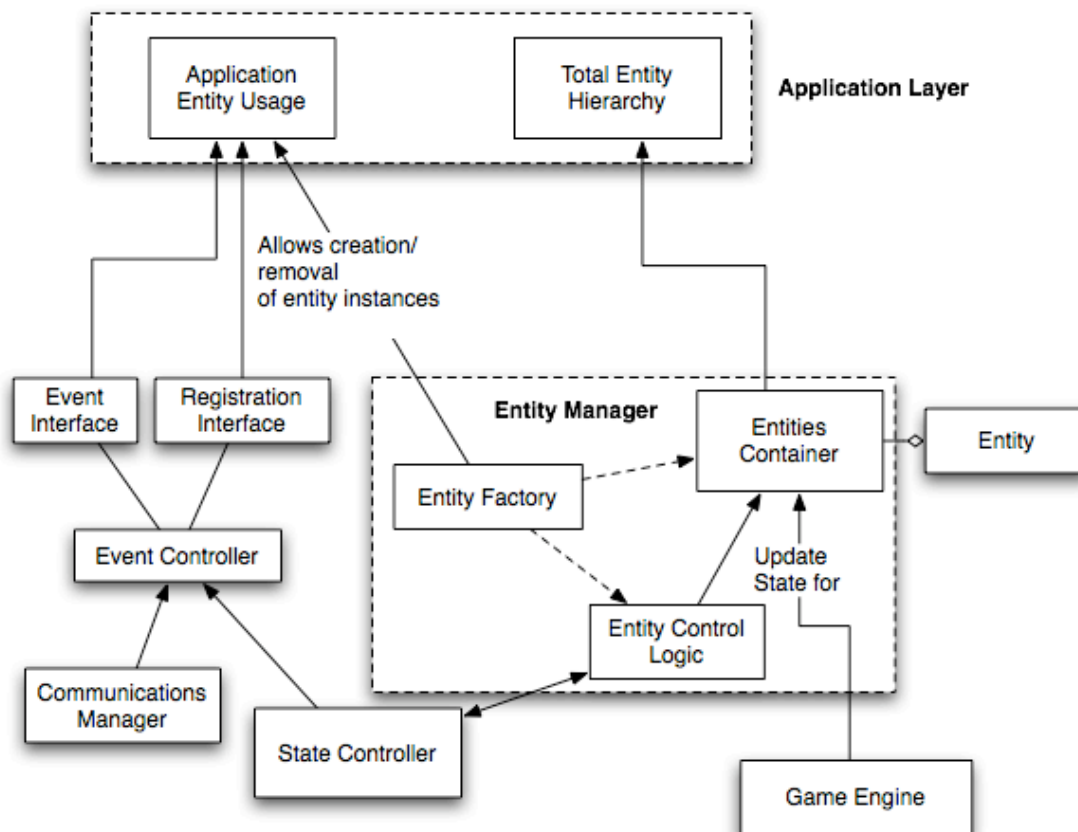


Responsibilities: Accepts updates from events, the game engine, and the map controller within the platform and sends rendering information to the application layer. Accepts updates from the display and updates state in the map controller and game engine.

Dependencies: Game engine, Map Controller, Event Controller  
Necessary for: Display, Map Controller  
Requirements Numbers: 1.2.\*

### 3. Entity Manager

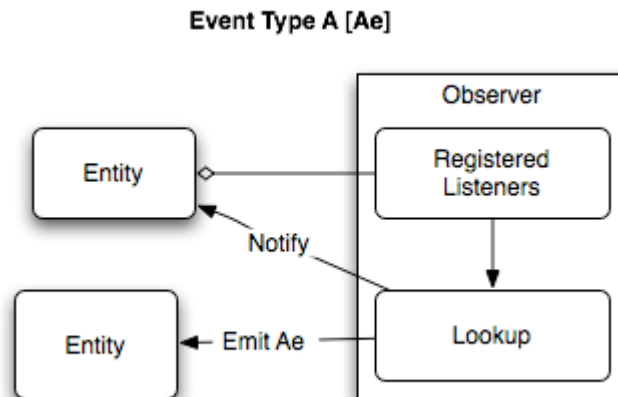
The entity manager is a data structure that holds information about each entity registered through the registration interface. The entity manager is necessary for the platform to effectively keep track of entities and notify them of events. The state controller and game engine will require frequent access 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.

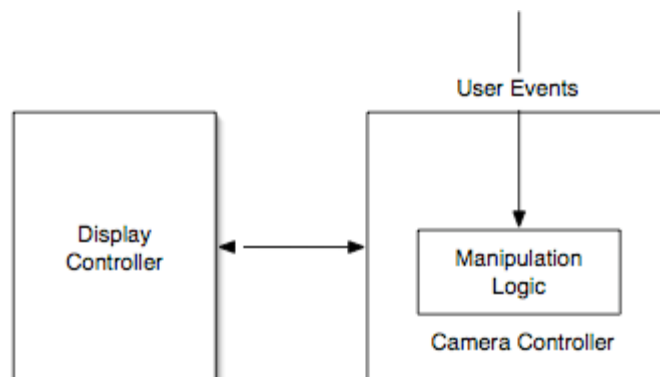
Dependencies: Application event creator, State Controller

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.



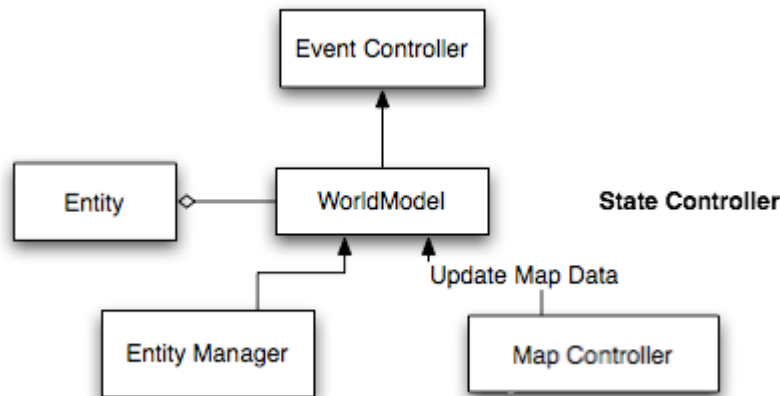
Responsibilities: Updates the camera properties through the display controller.

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

This is now fully managed by the GTGE engine, which we will be using in creating Pier436.

Responsibilities: Basic 2D sprites for application teams.  
Dependencies: None  
Necessary for: Extended Entity Hierarchy and Display rendering  
Requirements Numbers: 4.5

## 8. Base Entity Hierarchy

A set of interfaces and base classes will provide the application teams with an adequate API with which they can create entities for their own use.

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 control logic hierarchy begins with a set of interfaces which the application teams can implement in order to define their own control logic for their entities. The relationship between the entities and the control logic models the strategy pattern, such that any entity can be given a new control logic at any time. Entities may be swapped between different types of computer control, or perhaps from computer control to human control and vice versa.

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

### 10. Color Palette

Colors will be provided by the GTGE engine and the java standard API.

Responsibilities: Colors sprites on display.  
Dependencies: Sprite Library  
Necessary for: Display  
Requirements Numbers: 4.7

### 11. Message Protocol

A few messaging objects have been created to create a definite protocol. See the Appendix for details.

Responsibilities: Provide standard message format for application message passing.  
Dependencies: None  
Necessary for: Application message control  
Requirements Numbers: 4.6

### 12. Game Engine

GTGE and JGN will be used in Pier436. There are several advantages to this choice:

#### GTGE

- GTGE supports 2D graphics in a simple way.
- Sprites are available, and easy to wrap with our own data types.

#### JGN

- Supports message passing and synchronization between multiple instances of the GTGE game engine.

Responsibilities: Provides the graphical environment for viewing the application's world.  
Dependencies: Entity Manager, Display Controller  
Necessary for: None  
Requirements Numbers: 1.1.\*