

n8 Game Framework Design Document

Nate Ebel

February 28, 2014

Contents

1	Design and Use Patterns	3
1.1	Command	3
1.2	Observer	3
1.3	Subclass Sandbox	3
1.4	Singleton	3
1.5	Service Locator	3
1.6	Components	3
2	Client Use/Interaction	3
3	Resource Management	3
4	Game Loop	3
5	Window Management	3
6	Game States	3
7	Entity-Component System	4
7.1	Entities	4
7.2	Components	4
7.2.1	Component Communication	4
8	Entity States	5
9	Input	5
10	Events	5

11	Rendering	5
11.1	Rendering operations	5
11.2	Static sprite	5
11.3	Animated sprite	5
12	Camera	5
13	Operations on or using entities	5
14	Game levels	5
15	Globally Available Services	5
15.1	Logging	5
15.2	Audio Player	6
16	System wide values and enums	6
17	Configuration data/files	6
18	Use Cases	6
18.1	Input changes game state	6
18.2	Input changes player state	6
18.3	Input moves player	6
18.4	Entity destroyed and animation is played	6
18.5	Timed event activated	6
18.6	Timed event destroyed	6
18.7	Entity collision	6
18.8	AI player must make a decision	6
18.9	Notify an achievement system when 5 enemies are destroyed	6
18.10	Remap an action's key	6

1 Design and Use Patterns

1.1 Command

1.2 Observer

1.3 Subclass Sandbox

1.4 Singleton

1.5 Service Locator

1.6 Components

2 Client Use/Interaction

Client should instantiate a single instance of a Game object.

3 Resource Management

4 Game Loop

Takes place within Game and uses the Game object's GameTimer instance to control the loop. Within the loop the current GameState is processed at each iteration.

5 Window Management

Should be a window object. This object should be stored by the Game class. Should support renaming and resizing.

6 Game States

A GameState is whatever is happening in the forefront of the game. GameStates will be maintained on a stack within a GameStateManger. GameStateManager acts as a bridge between the game loop within Game and the currently active GameState. Should support passing of entities between states, and each GameState will maintain a list of its entities. GameStates will include 3 key methods of responding to user input, updating, and rendering.

The Update method will receive the current time as an argument, and Render will receive the current window canvas as an argument. These 3 methods will be called in the game loop within the Game object. Pause and resume methods can be overridden to handle changes to and from the GameState.

7 Entity-Component System

7.1 Entities

Entities will be constructed using an entity-component system. Entity will be a general class that can store different components. Different behaviors will then come from which components an entity has. Entity will have a broadcast method to send messages between components.

7.2 Components

Functionality can be added to entities through components. A component could be a position, a sprite, player information, or anything else that helps differentiate an entity or added to an entity's behavior. Components should take a GameObject pointer as an argument to its methods so a single instance of a component could potentially be used for many entities if the component is stateless.

7.2.1 Component Communication

7.2.1.1 Shared State Entity data that is cross-domain and will be used by multiple components should just be maintained in the entity class. This is a good candidate for data such as x and y position.

7.2.1.2 Direct Reference Components that are closely related could store direct references to one another. For now, this approach is not supported.

7.2.1.3 Messaging Components can broadcast and receive messages to one another. A component can then be wired to handle a particular message type. For example, physics component could detect a collision and broadcast a collision event which the audio component handles by playing a sound and the player info component handles by decreasing the player's health. These messages could be as simple as an int or they could be a message object containing more information.

8 Entity States

9 Input

10 Events

11 Rendering

11.1 Rendering operations

- Render single entity
- Render single entity with a camera object (Camera: see Section 12)

11.2 Static sprite

11.3 Animated sprite

12 Camera

13 Operations on or using entities

AI physics

14 Game levels

15 Globally Available Services

Use globally available service locator. The service locator should support:

- registering and unregistering of services
- should be initialized at start. This is when service objects will be provided to the locator

15.1 Logging

Logging should be available to any aspect of the system.

15.2 Audio Player

16 System wide values and enums

17 Configuration data/files

18 Use Cases

18.1 Input changes game state

18.2 Input changes player state

walking → flying

18.3 Input moves player

18.4 Entity destroyed and animation is played

18.5 Timed event activated

18.6 Timed event destroyed

18.7 Entity collision

18.8 AI player must make a decision

18.9 Notify an achievement system when 5 enemies are destroyed

18.10 Remap an action's key