5/20/2012

GROUP 19

SYSTEM DESIGN DOCUMENT FOR FALLOUT EQUESTRIA

Version: 2.0 | Lukas Kurtyan , Pontus Pall, Gustav Alm Rosenblad, Joakim Johansson

System design document for Fallout Equestria
Version: 2.0

Date: 5/20/2012

Authors:

Lukas Kurtyan

Pontus Pall

Gustav Alm Rosenblad

Joakim Johansson

This version overrides all previous versions.

Table of Contents

1 Introduction

1.1 Design goals

The project should be loosely connected in order to make anything possible.

1.2 Definitions, acronyms and abbreviations

- Archetype – A blue print for entities.
- Assets – Things that are loaded from hard drive, e.g. sound, textures, maps etc.
- Behavior – Defines how an entity behaves.
- Client – A computer connected to a host.
- Component – A class defining a property of an entity.
- Entity – A public key to a database containing components.
- Host - The computer, to which the other computers are connected. Acts as a server. The place where logic is done, and the game is run.
- Look and feel – A XML-file and a large texture containing all the normal backgrounds, highlights, etc. that make up the GUI.
- LWJGL – A graphics library for java enabling access to Open GL.
- GUI, graphical user interface
- Java, programming language
- Screen – A layer on the display.
- Game world – A world containing entity systems, an entity database and entities.

2 System design

2.1 Overview

The application uses a mix and match of different programming patterns and philosophies.

2.2 Screen systems

It is a layered system able to show several screens at once on top of each other. It uses different screens for ***GUI*** and ***game worlds***. These screens transition between each other according to player input.

2.3 Graphics system

All the graphics are done through ***LWJGL*.** SpriteBatch

2.4 Entity Framework

2.5 GUI

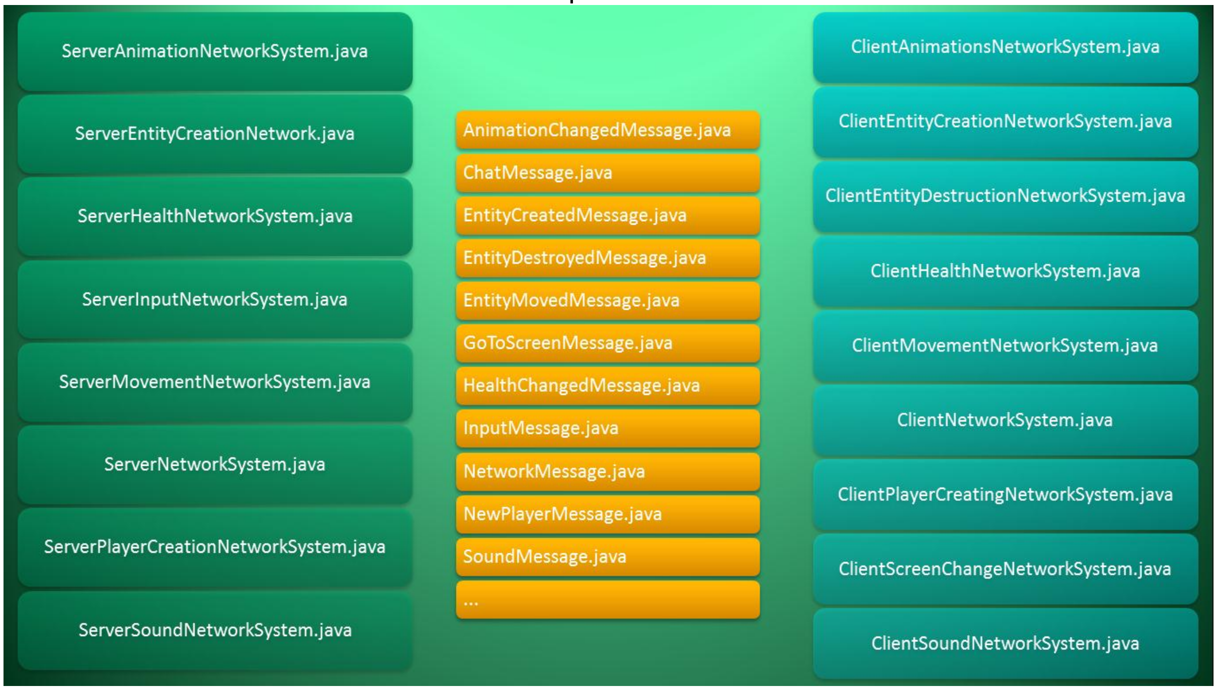
2.6 Screen

2.7 Network

The network uses the KryoNet library in order to send objects of the type NetworkMessage between the server and the clients. Both the server and the clients have network systems handling the incoming messages and sending new messages.

The architecture used is Authoritative Server, i.e. all the logic is done on the server side and then sent to the client side for rendering. The clients only send their input and the server does the rest.

To enable networking, use NetworkSystemBuilder and specify the world and other arguments needed by the the networked entity systems. Examples of functioning parts of the system is ServerMovementNetworkSystem (SMNS), ClientMovementNetworkSystem (CMNS), EntityMovedMessage (EMM). Each update, the server sends the position of all entities through an EMM. The client then receives the EMM and uses it to update its entities.


ServerAnimationNetworkSystem.java
ServerEntityCreationNetwork.java
ServerHealthNetworkSystem.java
ServerInputNetworkSystem.java
ServerMovementNetworkSystem.java
ServerNetworkSystem.java
ServerPlayerCreationNetworkSystem.java
ServerSoundNetworkSystem.java
AnimationChangedMessage.java
ChatMessage.java
EntityCreatedMessage.java
EntityDestroyedMessage.java
EntityMovedMessage.java
GoToScreenMessage.java
HealthChangedMessage.java
InputMessage.java
NetworkMessage.java
NewPlayerMessage.java
SoundMessage.java
...
ClientAnimationsNetworkSystem.java
ClientEntityCreationNetworkSystem.java
ClientEntityDestructionNetworkSystem.java
ClientHealthNetworkSystem.java
ClientMovementNetworkSystem.java
ClientNetworkSystem.java
ClientPlayerCreatingNetworkSystem.java
ClientScreenChangeNetworkSystem.java
ClientSoundNetworkSystem.java

2.8 Content

The framework is currently able to load files of the following formats:

- .anim (Format for animation)
- .animset (Format for a set of animations)
- .archetype (Format for entity archetypes)
- .effect (Format for vertex and fragment shaders)
- .font (Format for character information on the font texture)
- .ogg (Format for sound effects and music)
- .pchar (Format for player characteristics)
- .png (Format for textures)
- .scene (Format for the scenes in the game)
- .tdict (Format for look and feel, as well as body part information on animation textures)
- .XML (Format for things we didn’t come up with a good format name for)

Each format has one or more loaders derived from IContentLoader<T>, which converts the content into the type T. For XML-based content, JDom and XStream are used for the conversion. For loading of content, *contentManager.load(String path, Class<T> classToLoad)* is used. For example:

```
Texture2D texture = contentManager.load("foo",Texture2D.class);
```

2.9 Graphics

2.10 Behaviors

2.11 Animation

2.12 Scene

2.13 Utils

2.14 Software decomposition

2.14.1 General

Package diagram. For each package an UML class diagram in appendix

2.14.2 Layering

2.14.3 Dependency analysis

2.15 Concurrency issues

2.16 Persistent data management

2.17 Access control and security

2.18 Boundary conditions

3 References

APPENDIX