Spring 2007 – Saviour Developer Manual

Online Virtual Team Collaboration Platform

with 3D Graphics

Duygu ATILGAN 1394667
Nilgün DAĞ 1394857
Aslı ÖZAL 1514868
Bahadır ÖZDEMİR 1395342

June 15, 2008
# Table of Contents

1 Introduction........................................................................................................................................... 3

2 Architectural Overview and Basic Concepts.......................................................................................... 3

3 Data Directory Structure ...................................................................................................................... 3

3.1 Audio .................................................................................................................................................. 3

3.2 GUI .................................................................................................................................................... 4

3.3 SaviourMaterials............................................................................................................................... 4

3.4 SaviourModels................................................................................................................................. 4

3.5 Packs ................................................................................................................................................ 4

3.6 Particles ............................................................................................................................................ 4

3.7 SaviourTextures............................................................................................................................... 4

3.8 SaviourPaths.................................................................................................................................... 4

3.9 Objects ............................................................................................................................................. 4

4 Source Directory Structure ................................................................................................................ 5

4.1 Audio System................................................................................................................................. 5

4.2 Common.......................................................................................................................................... 5

4.3 Component System.......................................................................................................................... 5

4.4 Config System................................................................................................................................. 5

4.5 Game Listeners............................................................................................................................... 5

4.6 Game System................................................................................................................................... 5

4.7 Graphics System............................................................................................................................. 5

4.8 GUI System..................................................................................................................................... 6

4.9 Input System.................................................................................................................................. 6

4.10 Network System............................................................................................................................. 6

4.11 Object System............................................................................................................................... 6

4.12 Utility............................................................................................................................................. 6

5 Extending Game Logic.......................................................................................................................... 6

6 Extending the Map............................................................................................................................... 7

7 Extending the User Interface................................................................................................................ 7

8 Used Design Patterns.......................................................................................................................... 7

9 Used Libraries.................................................................................................................................... 8

10 Conclusion.......................................................................................................................................... 8
1 Introduction

This document is intended to be a complete reference manual for the simulation program, Saviour. It covers general concepts about the game engine internals, game data and the interaction between them. Also it briefly outlines the steps that need to be taken for enhancing the functionalities of the program. The modifications presented in this document require familiarity with XML files, Modeling, Graphics Programming and Network Programming.

2 Architectural Overview and Basic Concepts

Saviour simulation is an online virtual team collaboration imitating an abstract model of a real-life situation, evacuation after a disaster. Four users, which are medical unit, firefighting unit, search & rescue unit and the facilitator, use the program simultaneously. The simulation state at each user is expected to be same with the help of the networking.

The program is made up of mainly two parts: the game engine and the data files read and driven by the engine. The game engine is managed by the engine manager which controls and synchronizes the other managers. Main managers are created by the Engine Manager, at the beginning of the simulation. These managers, each of which has a thread running in the simulation concurrently, are responsible from managing and controlling basic tasks. At the main loop of the game, the game time is updated and the managers are ticked at each 1/60 sec. The tick wakes the sleeping manager threads, draws the next frame and processes the tasks in the task queue according to the scheduler algorithm. The game time is updated once per frame so that the calculations are consistent across the managers. To ensure that the rendering is done fast the graphics thread is run whenever possible and it will use more CPU against the other managers. The other threads, which are audio thread, network thread, input thread and object managing thread, are run only at each tick. The tasks in the task queue, which correspond to the network message handling functions, are also processed at each tick.

The game logic of the Saviour is mainly dependent on the Object – Component System approach. In this approach, the data object hierarchy is flattened down to a single data object class. This class aggregates some number of components each of which supports a different functionality. In other words, each component is a self-contained piece of program logic. Thus by assembling components into objects, each type of object owns different functionalities. In this way, functionality is forced out of the objects itself to components. With the help of object-component system, new type of objects can be generated easily either from the existing component types or from newly generated component types.

3 Data Directory Structure

In Saviour, all simulation data and resources are in Media directory. Under this directory, there are subdirectories which are as follows:

3.1 Audio

This directory contains the audio files played at any instance of the game. Any audio file to be added should be put under this directory.
3.2 GUI

This directory contains the files needed for the GUI implementation. Font files, imagesets, .jpg files included in the GUIs, layouts, scheme, tga files and look’n’feel files are saved under this directory.

3.3 SaviourMaterials

This directory contains the material files needed for the models. A material script expresses attributes of the model’s appearance, such as lighting and texturing. The use of material scripts bring the ability to define complex materials in a script which can be reused easily. Material scripts are loaded when resource groups are initialized. Ogre looks in all resource locations associated with the group for files with the '.material' extension and parses them.

3.4 SaviourModels

This directory contains the models, i.e. the mesh files in the Saviour. As Saviour is divided into areas, each area’s own models are saved in a separate directory for obtaining simplicity. There are four subdirectories for the models which are General, Engineering Area, Mall Area and Dormitories Area.

3.5 Packs

This directory contains the Ogre overlays which are shown in the scene throughout the simulation.

3.6 Particles

This directory contains the .particle files needed for fire creation in the simulation.

3.7 SaviourTextures

This directory contains the textures, i.e. .jpg files used in the models.

3.8 SaviourPaths

This directory contains the path files needed for moving animation of fire trucks, ambulances and other vehicles. Those paths are from the teleport points to the each building in the areas.

3.9 Objects

This directory contains the all of the data related to objects in the simulation. It contains five subdirectories which are General, Mall Area, Dormitories Area, Engineering Area and Resources. In the General subdirectory there is ObjectTemplates file explaining which objects are created from which components. In the directories related to areas, there are xml files describing each building and each civilian in that area. In the Resources, subdirectory , resource objects are described in separate xml files. These xml files can be examined to understand the creation of the objects from the components.
4 Source Directory Structure

Source code of Saviour is in two directories in CVS which are src and include. However, the directory structure is organized in the project’s .vcproj file which is an xml file including project settings and references to files used by the project. It serves as a central file that organizes the project files. So, in the Saviour, there are the filters below for organizing the source code:

4.1 Audio System

The source code under this filter is responsible from playing audio files in the simulation, playing audio coming from network, and recording audio from the user. SaviourAudioCapture records the user’s voice using the OpenAL functions. SaviourAudioContext creates OpenAL’s default audio context and handles the audio sources. SaviourAudioManager manages the audio thread. SaviourAudioEncoder encodes the captured data using Speex library. Lastly, SaviourAudioSource plays audio from the audio sources.

4.2 Common

Under this filter, in SaviourCommon, there are type and constant definitions. In SaviourPrerequisites there are class declarations for forward declaration. In SaviourConfig, there is the code for assertion handling.

4.3 Component System

Under this filter, there are component classes related with the game logic.

4.4 Config System

Under this filter, ConfigManager and ConfigElement classes handle the configuration of the other managers.

4.5 Game Listeners

Under this filter, there are classes related to event listening. TickListener is the class for the tasks that need to be done at each tick. InputListener is the class for listening mouse and keyboard inputs using OIS library. NetworkListener is the class for handling the Network messages.

4.6 Game System

Under this filter, there are classes related to game logic. EngineManager, manages the other manager threads. GameManager handles the game logic.

4.7 Graphics System

3D Graphics of the simulation is handled in this system using Ogre library. GraphicsManager manages the graphics thread. SceneManager is used for abstraction of the Ogre Scene Manager. GraphicsResourceManager is responsible from initializing, loading and unloading of the Ogre resource groups. RenderTargetManager manages the camera, viewport and rendering window settings. Finally, overlays in simulation also belong to the overlay classes under this filter.
4.8 GUI System

In this directory, GuiManager class handles the user interface related functions. Other than that, in game interface related classes for each user are contained in Gui System.

4.9 Input System

Under this filter, there are classes related to input listening. InputManager handles the input managing thread. The classes such as KeyboardDevice, MouseDevice and InputEventHandler is used for the abstraction of the OIS library functions and classes.

4.10 Network System

Under this filter, there are classes related to network implementation of the simulation.

4.11 Object System

Under this filter, there are classes related to object creation and object loading from the xml files and object managing.

4.12 Utility

Classes under this filter are used for several utilities used in the Saviour. Under the Log System subfilter, there are classes needed to save the Log files. Under Message subfilter, there are classes related to Messages in the Saviour which are used for converting the packets to messages and inserting those messages into the scheduler. Under Scheduler subfilter, there are classes related to scheduling and processing of the messages. Also classes related to exception, clock and timer are under Utility filter.

5 Extending Game Logic

Game logic of Saviour can be extended by both adding new object types and adding new functionalities to the existing objects. The best way to add new functionality is adding new Component classes. Objects including the new components gain new functionality in this way.

For a new object to be created first a new object template should be added to the object templates file.

Below is a part of the ObjectTemplates.xml file which contains the Firefighter object type made up of three components.

```xml
<?xml version="1.0"?>
<objectTemplates>
  <objectTemplate type="Firefighter">
    <component type="ComponentVisual"/>
    <component type="ComponentFirefighterResource"/>
    <component type="ComponentActor"/>
  </objectTemplate>
  ...
</objectTemplates>
```
For example if a new object type such as Police is needed to be added, it would be enough to add a new component such as ComponentPoliceResource to the component types and combine it with the existing components to create the object Police. There are ten object types in total which are Firefighter, SearchRescue, Medical, Facilitator, Building, FirefighterResource, MedicalResource, SearchRescueResource, Civilian, Vehicle, and Visual. And there are ten component types which are:

- **ComponentActor** for user’s unique properties such as score,
- **ComponentAudio** for objects playing sound,
- **ComponentBuilding** for building properties,
- **ComponentFire** for representing fires in buildings,
- **ComponentFirefighterResource** for Firefighter actor’s resources,
- **ComponentMedicalResource** for Medical actor’s resources,
- **ComponentSearchRescueResource** for Search&Rescue actor’s resources,
- **ComponentVisual** for the object’s having a graphical representation in the simulation,
- **ComponentHealth** for Civilian’s properties,
- **ComponentMoveAnimation** for objects’ moving animation.

6   Extending the Map

In Saviour, there are areas each of which corresponds to a part of METU Campus. For now, three areas of the campus are implemented which are Dormitories, Engineering and Mall Area. Areas are added according to the METU map file in 3ds format. If a new area is to be added, firstly the area should be created as a scene in 3dsMax compatible with the map. After that, the 3ds objects in the scene should be converted to mesh files and the .scene file should be created. DotScene file is an xml file which includes the scene nodes descriptions correspondent with the scene nodes in Ogre.

7   Extending the User Interface

In order to add new features to user interface, hands on experience with CEGUI is needed. Graphical user interface definitions are read from the layout files under the Media/GUI. Event handling related to the user interface is done in the classes under GUI System filter.

8   Used Design Patterns

- **Factory Method**: Used in MessageFactory class for creation of the network messages.
- **Singleton**: Used in all manager classes for restriction of the class to one object.
- **Facade**: Used in EngineManager class for providing a simplified interface to the remaining source code.
- **State**: Used in Network System for changing the NetworkEngine object’s type at runtime. Network Engine could either set to be Client or Server after the game starts with the help of this pattern.
- **Observer**: Used in Listener classes. The events addressed by Observer pattern are handled in the listener class’s functions.
- **Strategy**: Used in Scheduler class for choosing one of the algorithms that is implemented for message processing in the scheduler.
- **Command**: Used in network system for event handling actions. Each command object which corresponds to a message inserted into scheduler encapsulates the action related to handling that message.
Template: Used in Manager and Component classes. It allows code to be written without consideration of the data type with which it will eventually be used.

9 Used Libraries

Saviour is mostly developed on open source solutions. Also in this way, extending the simulation engine will be easier when the libraries are extended and updated. In Saviour, Graphics is based on OGRE (www.ogre3d.org). Audio is based on OpenAL (www.openal.org). GUI system is based on Crazy Eddie’s GUI System (www.cegui.co.uk). Modeling and building the map is done with 3ds Max 2008 (usa.autodesk.com). Exporting of the 3ds files to mesh files id done with OgreMax Exporter plugin for 3ds Max. Xml parsing is done with TinyXML (www.grinninglizard.com/tinyxml) and finally Network system is based on RakNet (www.jenkinssoftware.com).

10 Conclusion

By the end of the second semester, most of the features planned in the Detailed Design Document have been implemented in Saviour. Implementation of network system, audio system, game logic, and graphics except 3D animation and illumination has been completed. Since design decisions haven’t changed much and implementation has been done in correspondent with the design document, it has been completed with few problems.