PUBLISHED BY

Electrical Engineering Department

Faculty of Industrial Technology

Institut Teknologi Sepuluh Nopember (ITS)

ITS Campus, Keputih, Sukolilo

Surabaya 60111

ISSN 2355-8636

The 15th Seminar on Intelligent Technology and Its Applications

Copyright © 2014 by Electrical Engineering Department, Institut Teknologi Sepuluh Nopember

All rights reserved. No part of this book, may be reproduced, stored, or transmitted, in any forms or by any means without the prior permission in writing from the publisher.

May, 22nd 2014

Dept. of Electrical Engineering Building,
Institut Teknologi Sepuluh Nopember,
Surabaya, Indonesia

Supply Chain Food Management using HFSM for RTS Game

Dihin Muriyatmoko¹⁾, Mochamad Hariadi²⁾

Department of Electrical Engineering, Faculty of Industrial Technology
ITS Surabaya Indonesia 60111, email: dihin13@mhs.ee.its.ac.id

ABSTRACT

Finite state machines have become extremely popular over the last decade and helped game developers build some pretty fun RTS games [1]. Finite State Machines have been widely used as a tool for developing RTS games, especially as pertains to solving problems related to AI, input handling, and game progression [2]. Anyhow, to control game play and user interface for Supply Chain Management (SCM) of Food on RTS game only played conventional finite state machine (FSM) design. Therefore in this research is developed Food SCM using Hierarchical State Finite Machine (HFSM).HFSM allow for a modular development of states that is more maintainable and scalable[3]. The formalism of HFSM makes the state machine approach truly applicable to real-life embedded systems [4].

Keywords: SCM, FSM, HFSM, UI, Game Play
Food SCM

CHAPTER 1: INTRODUCTION

As gamers and game developers we know that immersion is everything. When you're immersed you lose track of time and become involved in what the game is presenting. A major factor in what makes or breaks immersion is how easy it is for your player to convert an idea into an in-game action that is, how fluid your game's User Experience (UX) is and how well-designed its User Interface (UI) is. A game hurts itself by providing too little information or too much, requiring too many inputs, confusing the player with unhelpful prompts or making it hard for a new player to interact. Poor UI design can even break your game completely [5].User Interface and UX in game development in this experience is how to control food supply chain management using hierarchical state finite machine.

CHAPTER 2: FUNDAMENTALS

What Are UI and UX?

The terms UI and UX are sometimes (incorrectly) used interchangeably, but they have specific meanings. UI, or User Interface, refers to the methods (keyboard control, mouse control) and interfaces (inventory screen, map screen) through which a user interacts with your game. UX, or User Experience, refers to how intuitive and enjoyable those interactions are.

Who Should Design a Game's UI?

Many programmers are scared off from making UIs because they feel that this is a creative job, one better left to someone like an artist who has a better understand of presentation. And yet UI design is a

largely logical process, one that's perfectly approachable to a developer.

In fact, I believe that the programmer (or creative director, or whoever in your team has the most direct role in shaping the playing experience of your game) should be responsible for making the UI, as they are the ones who know the game inside and out, and who know what information is important and what is incidental.

They should focus on the functional aspects of the UI: How big is it, does it (or should it) scroll, what information is displayed and where, and how the player navigates through it. Mock it up with two or three colors, and never rely on color changes alone to convey information. At the very least, run it through ColorOracle or Coblis to ensure that color-blind people aren't missing out.

Games with Poor UIs

Before getting into this, it's important to understand that when I talk about bad UIs, I make that judgment based on the ideas and behaviors that I bring to it as a PC gamer who plays games with a keyboard and mouse. The user's choice of input is extremely relevant to the user experience, although it's not the whole story. Someone who is used to a touch interface is going to find a button interface awkward, for example, but a well-designed button interface will still be easy to understand and predict.

Oblivion

Oblivion's UI is a classic example of one platform's UI being incompatible with another. Even though Oblivion was ported from consoles to PC, it still retained its controller-focused UI. This is completely analogous to releasing a game on iOS or Android and making the player move an on-screen cursor around like they're using a laptop's track pad instead of taking advantage of absolute touch positioning. So what makes Oblivion's UI so bad? It's a combination of wasted space, improper scaling, and inappropriate controls.

Figure 1: Oblivion User Interface of Screen
Inventory

I hope you're cool with viewing six items at a time; these windows aren't adjustable. The actual part of the menu that conveys the relevant information, such as the contents of your inventory, uses only 21.7% of the total screen space at a resolution of 1920x1080.

Figure 2: Oblivion User Interface of Inventory only used 21,7% of Screen Space

Games with Good UIs

But let's not spend all this time talking about bad UIs. Great examples of UIs come from games which acknowledge the limitations and strengths of their systems.

Morrowind

Most PC gamers agree that Morrowind (Oblivion's predecessor) has one of the best PC UIs around - except for one issue which I'll talk about below.

Figure 3: Morrowind User Interface Screen of Inventory

Let's go over the basic elements of this screen.

- It's a single screen with four different, untapped windows. Clockwise from top-left you have the character details, the map (which you can zoom), your list of spells and magic items, and your inventory (with tabs for sorting items).
- At the top-right of each of these windows is a small raised square. If you click this square, then that particular window will remain on-

screen even after you exit Menu mode. You can keep the map window open if you're trying to completely map the coast, for example, or you can keep the Magic window open to keep an eye on the number of charges left in your Ring of Healing.

- You can drag these windows anywhere on the screen, and you can resize them or minimize them to tailor the UI to your needs.
- At the bottom-right of the screen are some status icons. If you see an unfamiliar icon there you can hover over it in Menu mode, and a tooltip will tell you what it is. The same icons appear at the top of the Magic window too.
- The inventory is laid out in a space-efficient grid, and you can adjust the number of items it displays by resizing the inventory screen.

What does this UI mean for the PC player? Almost everything is *literally* one click away. If you want to look at your skills you scroll the character window. If you want to equip something, you drag it onto your paper doll. The interface is fast and accessible, and almost every type of item in the world has a unique inventory icon, so finding your Journeyman Hammer involves half a second of looking at your neat, scalable inventory.

The only detail that lets this UI down is the fact that potions and scrolls reuse inventory icons; there are unique icons for the quality of the potion, but not for its effect, so a Master Water Breathing potion looks just like a Master Restore Fatigue potion[5].

What is a Supply Chain

A network of retailers, distributors, transporters, storage facilities and suppliers that participate in the sale, delivery and production of a particular product. So, a supply chain is product specific, not company specific.

Figure 4: Network of Supply Chain

Supply Chain Management (SCM)

"Supply chain management involves the flows of material, information and finance in a network consisting of customers, suppliers, manufacturers, and distributors" (Lee, 2000)

SCM is the science and art of making sure the right stuff is at the right place at right time to lower

costs, and to create value. Most of this stuff is physical stuff. But also, information and money have to be at right place at right time. As diagram shows, supply chains are allowed about flows. Most of flow comes from suppliers to customers (not surprisingly). But are reverse flow to suppliers from customers

There are three types of flows:

1. Most obviously, is flow of actual materials, the middle bars
 - From suppliers: flows of raw materials, intermediate products, finished goods
 - Reverse material flows: returns, repairs, servicing, recycling, disposal
2. Beyond actual materials, there are information flows
 - From suppliers: manufacturing capacity, delivery schedules, promotions they have going
 - Reverse flows: sales, orders, inventory, quality, promotions
3. And finally, there are financial flows:
 - From suppliers: Credits, consignment, payment terms, invoice
 - Reverse Flows: payments, consignment

Effective SCM means keeping all these flows humming, efficiently and accurately [6]

Figure 5: Flows of Supply Chain Management

Finite State Machine (FSM)

A finite-state machine (FSM) or finite-state automaton (plural : automata), or simply a state machine, is a mathematical model of computation used to design both computer programs and sequential logic circuits. It is conceived as an abstract machine that can be in one of a finite number of states. The machine is in only one state at a time; the state it is in at any given time is called the current state. It can change from one state to another when initiated by a triggering event or condition; this is called a transition. A particular FSM is defined by a list of its states, and the triggering condition for each transition.

The behavior of state machines can be observed in many devices in modern society which perform a predetermined sequence of actions depending on a sequence of events with which they are presented.

Simple examples are vending machines which dispense products when the proper combination of coins is deposited, elevators which drop riders off at upper floors before going down, traffic lights which change sequence when cars are waiting, and combination locks which require the input of combination numbers in the proper order.

Finite-state machines can model a large number of problems, among which are electronic design automation, communication protocol design, language parsing and other engineering applications. In biology and artificial intelligence research, state machines or hierarchies of state machines have been used to describe neurological systems and in linguistics to describe the grammars of natural languages.

Considered as an abstract model of computation, the finite state machine is weak; it has less computational power than some other models of computation such as the Turing machine [7].

FSM is a technique that appears in many different forms and is used in major game titles such as Quake, Fifa Soccer, and Warcraft. In fact, FSM is the most popular approach for AI algorithms used in games [8]

Hierarchical Finite State Machine (HFSM)

The Finite State Machine (FSM) is a design pattern in which actions are determined by events and the current context of the system. The driver code dispatches events to the FSM that forwards it to the current state. Functions processing the events decide what should be the next system state. After that, the process repeats. The Hierarchical FSM is an extension of the FSM concept. In this case, any state can be a sub state of some larger state[9].

Most developers say they use a "HFSM" in their games. But you'll be surprised how different their technology can be despite them using the same name to describe it. There are two reasons why the HFSM is used?

1. To provide ways to re-use logic by sharing external transitions.
2. To help split up the problem internally into sub parts.

Granted, there's a certain overlap between these two concepts, but yet they are orthogonal issues (you can do one without doing the other). Here are two opposing examples, one FSM for animation and the other for AI behaviors.

Using Groups of States to Share Transitions

Say you want to build a finite state machine for an animation system. In this case, you need:

- Complete control over every low-level state in the system, so you can make sure the correct animation is played in every case.
- A set of tools to help you reuse logic so that you don't have too much redundancy in the transitions.

To build this kind of FSM, you do the following:

1. You build the system as a **flat state machine** by default, only grouping states together to reuse transition logic.
2. You follow the process of reusing logic by “**tagging**,” e.g. you can dive/sprint/crouch from this state.
3. You think from the **bottom-up** deciding which transitions each state should inherit.
4. You always edit the FSM **as a whole** to keep the big picture, as much of the system doesn’t make sense when broken down into small parts.

When you do this, the hierarchy becomes loose collection of states used informally to reduce redundant logic by sharing transitions [10].

Figure 6: HFSM Using Groups of States to Share Transitions

Hierarchical State Machine Example: The following state transition diagram recasts the state machine by introducing two levels in the hierarchy. In service and Out_Of_Service are the high level states that capture the common message handling. Active and Standby states are low level states inheriting from in service state. Suspect and Failed are low level states inheriting from Out_Of_Service state.

Figure 7: HFSM example diagram

The following diagram clearly illustrates the state hierarchy. Even the in service and Out_Of_Service, high level states inherit from the Unit State that is at the highest level [11].

Figure 8: Illustrates of state hierarchy Diagram

CHAPTER 3: METHODES AND DESIGNS

In this research is developed Food SCM using hierarchical state finite machine (HFSM).

Figure 9: Block Diagram Experiment of food SCM using HFSM

The game is played in represents a supply chain in which there are 5 roles: Grower, Processor, Wholesaler, Retailer, and a Customer.

Figure 10: Design of Food SCM Flow

SCM Flows

If the goal of SCM is to provide high product availability through efficient and timely fulfillment of customer demand, then how is the goal accomplished? Obviously, you need effective flows of products from the point of origin to the point of consumption. But there's more to it. Consider the diagram of the fresh food supply chain. A two-way flow of information and data between the supply chain participants creates visibility of demand and fast detection of problems. Both are needed by supply chain managers to make good decisions regarding what to buy, make, and move.

Other flows are also important. In their roles as suppliers, companies have a vested interest in financial flows. As you can understand, suppliers want to get paid for their products and services as soon as possible and with minimal hassle. Sometimes, it is also necessary to move products back through the supply chain for returns, repairs, recycling, or disposal [12].

By that scenario of food SCM we can make concept a simple hierarchical food SCM like on this design

Figure 11: Flow Food SCM using HFSM

By that flow we can make a flow diagram food SCM using HFSM on the RTS game like this

Figure 12: Flow Food SCM using HFSM on RTS Game

Farmer is simplification to Grower, Restaurant to Processor, Porter Cart to Wholesaler / Retailer, Troops Camp to customers and Rats is Enemy.

The game is likened a kingdom/Empire. We must run it like an owner of kingdom. The plan of game is play as a manager of Farmer, Restaurant, Porter Cart, and Troops Camp and Fights the Rats.

CHAPTER 4: EXPERIMENT

In this experiment we can see the plan of RTS game applied as a manager (king) of kingdom.

The rule of Food SCM on this RTS game as following here:

1. The kingdom is poor and want to be rich
2. The way to be rich must colonize another kingdom, others kingdom only defends and not attack our kingdom
3. The kingdom have Fife recourses
 - a. Farm is to plant for supply food matter
 - b. Restaurant is to cook the food matter and save temporary food
 - c. Porter Cart was delivered food to Troops Camp to supply food for Troops
 - d. Troops Camp was accommodate the Troop and have parameter Food Reserve and need supply food from Porter Cart
 - e. Troops are to attack another kingdom and find Gold to build our kingdom.
4. The Enemy or Disturber is Rats and wants to thief the food Matter from Farm and we must fight them by some touch the rats. The rats is appear randomize depend the capacity of Farm
5. After the Troops getting Gold, the kingdom can upgrade their resources (Farm, Restaurant, Porter Cart, Troops Camp and Troops)
6. To detect the health of Troops is shown the health progress bar
7. To detect how to deliver food on Porter Cart is shown the deliver progress bar
8. To detect food reserve on Troops Camp is shown the reserve progress bar
9. The Farm is product food by automatically continue and have timer parameter to shown when the food matter is ready to harvest, if the food matter was take out of timer, it was gone
10. The Farm have capacity parameter to shown that food matter is full or not
11. The Restaurant have timer parameter to shown that the food is ready to save and take by Porter Cart, if the food was taken out of timer, it was spoiled and gone
12. The Restaurant is cooking food by automatically when the Food matter on Farm was touched and have capacity parameter to shown that food is full or not
13. The Porter Cart have timer parameter to shown how it deliver food and the time to deliver it is depend by how long the distance between kingdom to another's
14. The Porter Cart have capacity parameter to shown that food is full or not
15. The Troops Camp have capacity parameter to shown that food is full or not
16. The Troops Camp can accommodate limited troops depend the capacity
17. Flow food matter from farm to restaurant is automatically delivered

By that rule we can draw detail of Food SCM using HFSM on RTS game like this

Figure 13: Food SCM using HFSM on RTS Game

CHAPTER 5: SUMMARY AND THE ROAD AHEAD

By all experiment we can give conclusion that this game is how to manage not only food supply chain on RTS game but this manage the gold and health information like on this table

Resources	Behavior	Flows
Farm	Plant, Harvest, Fight	Food Matter
Restaurant	Cook, Save	Food
Porter Cart	Take, Deliver	Food
Troops Camp	Save, share to Troops	Food
Troops	Attack, Eat food, Healthy	Gold, Healthy info

Table 1: Manage Resources of behaviors and flows

Resources	Parameter	Upgrade By Gold
Farm	Timer, Capacity	Capacity
Restaurant	Timer, Capacity	Capacity
Porter Cart	Timer, Capacity	Capacity
Troops Camp	Capacity	Capacity
Troops A, B, C	Health/Energy	Strength

Table 2: Manage Resources of Parameters and Upgrade by Gold

REFERENCES

- [1] <http://aigamedev.com/open/article/fsm-age-is-over/>
- [2] William Mallouk, Esteban Clua, AN OBJECT-ORIENTED APPROACH FOR HIERARCHICAL STATE MACHINES, Departamento de Informática-PUC-Rio Vision Lab
- [3] <http://aiandu.blogspot.com/>
- [4] <http://www.barrgroup.com/Embedded-Systems/How-To/Introduction-Hierarchical-State-Machines>
- [5] <http://gamedevelopment.tutsplus.com/tutorials/game-ui-by-example-a-crash-course-in-the-good-and-the-bad--gamedev-3943>
- [6] Robert G. Fichman, MI 703 CIS Session #5, – MI 703 Computer Information Systems, 02/21/07
- [7] http://en.wikipedia.org/wiki/Finite-state_machine
- [8] Baillie, P. Programming Believable Characters in Computer Games, pp. 234-243
- [9] <http://www.drdobbs.com/hierarchical-state-machine-design-in-c/184402040>
- [10] Alex J. Champandard on October 21, 2007, "Choosing a Hierarchical FSM or a Hierarchy of Nested FSMs?" <http://aigamedev.com/open/article/hierarchical-or-nested-fsm/>
- [11] <http://www.eventhelix.com/realtime mantra/hierarchicalstatemachine.htm#.U1tmTFdpcxw>
- [12] <http://www.careersinsupplychain.org/what-is-scm/flows.asp>