radical entertainment

Technical Design

Document Number:	000xx	**Document Issue:**	0.1
Date Issued:	October 29, 2001	**Document Status:**	Draft
Filing Path:	//depot/Mirth/documents/TechnicalDesign/FETechDesign.doc		
Document Title:	Mirth Front End Technical Design Document		
Author:	Lorraine Cobb		
Keywords:	Mirth, GUI, Front-End		

Revision History

Issue	Date	Person	Reason
0.1	Nov 26, 2001	Lorraine Cobb	Initial draft for internal review.

1. INTRODUCTION

This document describes the technical design of the Mirth game's Front End.

1.1 Scope

This document outlines the design envisioned during the Pre-Production phase so that it can be used to aide in coding the GUI.

1.2 Objectives

To document the planned design of the Front End.

1.3 Intended Audience

This document is mainly intended for the Technical Director and Developers working on the Mirth game.

© Radical Entertainment. All rights reserved. No part of this publication, or any software included with it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, including photocopying, electronic, mechanical, recording or otherwise, without prior written permission of the copyright holder.

This document contains proprietary information of Radical Entertainment. The contents are confidential and any disclosure to persons other than the officers, employees, agents, or subcontractors of the owner or licensee of this document, without prior written consent of Radical Entertainment is strictly prohibited.

1.4 References

[1] Pete Low, Jacob Krarup; [SV080:1668]//depot/Mirth/documents/GameDesign/Design/Design Doc.doc - *Monster Ball Game Design*

[2] Lorraine Cobb; [SV080:1668]//depot/Mirth/documents/TechnicalDesign/MirthArch.doc – *Mirth Game Software Architecture Overview*

[3] Darren Esau; [SV080:1668] //depot/Mirth/libraries/radmovie/doc/Foundation Movie Player Interface.doc - *Foundation Movie Player Interface Detailed Design Document*

1.5 Acronyms

FMV	Full-Screen Movie
FTT	Foundation Tools and Technology
GUI	Graphical User Interface
HUD	Heads-Up Display
NIS	Non-Interactive Sequence
RGB	

This Document contains Confidential Information of Radical Entertainment.

2. TECHNICAL DESIGN OVERVIEW

2.1 Overview

The Front End System is responsible for managing the Graphical User Interfaces and various FMV movies (are NIS movies handled too?) displayed throughout the game. The GUI includes various pre-game menu option screens, a HUD overlay, a pause screen menu, and a game over screen. Flow charts and storyboards outlining the various GUI displays can be found in [1] The art that makes up these views are put together and stored in a Scrooby Project built by the RGB team. There will be four Scrooby projects for this game and they are defined as follows:

- PreGamePlay Project: consists of the Title Screen and all of the game option selection screens presented prior to game-play and therefore loaded during pre-play game state.
- InGamePlayPlay Project: contains all the types of HUDs which are active during the game as well as a Pause Menu screen and therefore loaded during game-play and pause game states.
- PostGamePlay Project: contains all the possible Results and Game Over Menu screens and therefore loaded during post-play game state.
- GameLoad Project: contains a Loading Screen to be displayed during state transitions. This project should be loaded at all times.

The FMV movies are supplied in .avi format. There is 5 FMVs for each of the companies involved in this game, namely, Nintendo, Pixar, Disney, THQ and Radical. There is also a demo FMV, and an Intro and Outtro FMV for each Gameplay.

Figure 1 is a class diagram of the Front End System. The main class that oversees everything is the FESystem class. It contains an instance of the Scrooby singleton class ‘App’. A pointer to the Scrooby ‘App’ is accessed during initialization of the FESystem at the beginning of the game. The Scrooby App object is used to access, load, unload, and draw Scrooby projects. Whenever a Scrooby project is loaded, a new project manager is instantiated to handle project visual update requests sent to the FESystem from the FEUserInputHandler in response to user controller input. The three project managers are the PreGamePlayGuiManager, the InGamePlayPlayGuiManager, and the PostGamePlayGuiManager – all owned by the FESystem. The FESystem passes the current project pointer to it’s Project Manager when it instantiates it. The FESystem also has a pointer to the Scrooby project that is currently active, and one pointer that always points to a ‘Loading’ project that remains open throughout the game and is displayed during game state transitions.

This Document contains Confidential Information of Radical Entertainment.

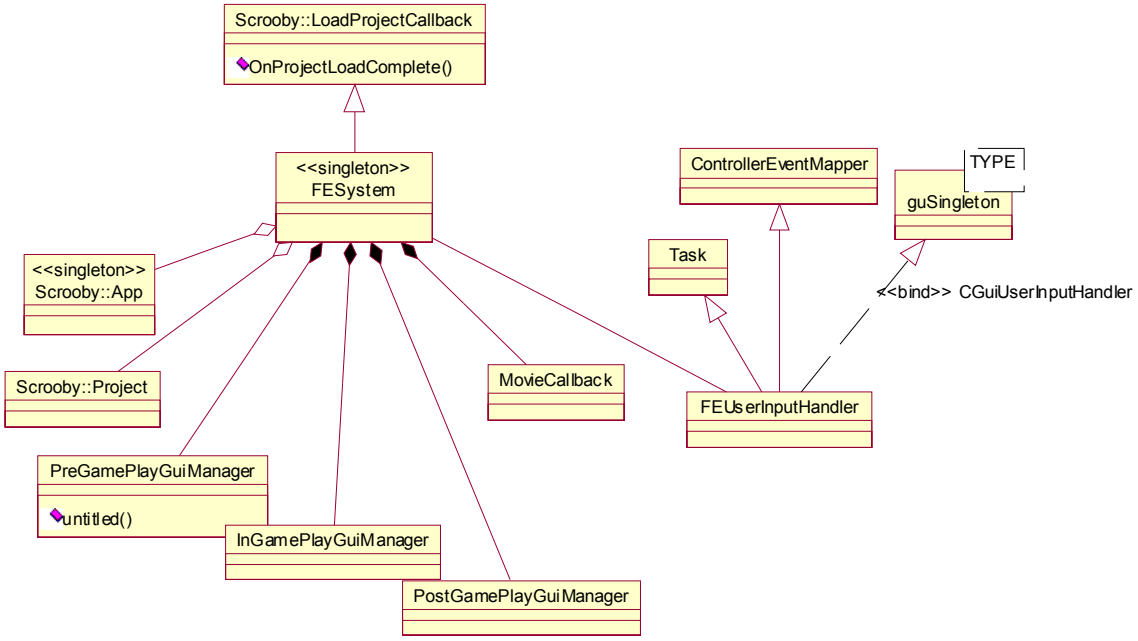

Figure 1 Front End System Overview

Figure 2 details an example of one of the Project Managers, namely the Pre-Game-Play GUI Manager. A manager typically owns a pointer which points to the Scrooby Project it is managing. It may also have pointers to other various Scrooby Project elements.

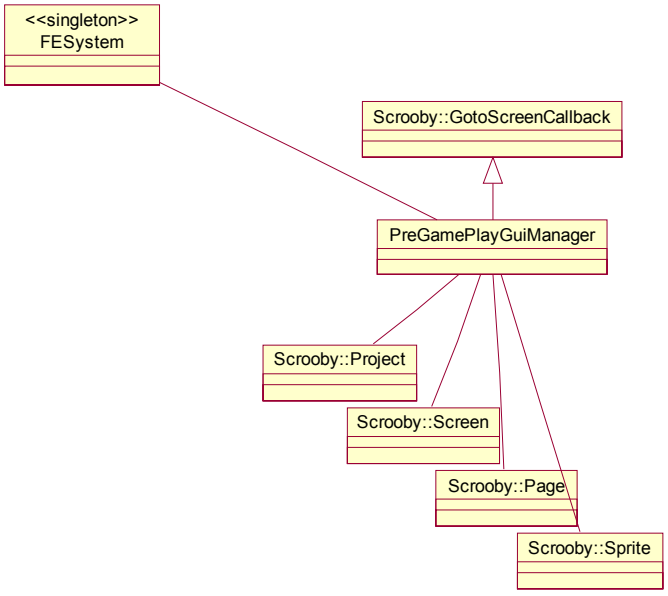

Figure 2 Pre-Game-Play GUI Manager

This Document contains Confidential Information of Radical Entertainment.

The FMV movies are launched by a system implemented using Foundation Tech's Movie Player system (radmovie library). A MovieCallback class is inherited from radmovie's IRadMoviePlayerCallback class. This provides the OnEvent method that is called when a FMV movie has finished playing. It also contains a pointer to a RadMovieSrceenPlayer object. This controls the Loading, Playing and Stopping of the FMV movies, and calls the MovieCallback's OnEvent() method. Figure 3 expands the implementation of the Front End's MovieCallback class to show the radmovie components it uses and contains. The remainder of this document references these classes and others illustrating various aspects using UML notation.

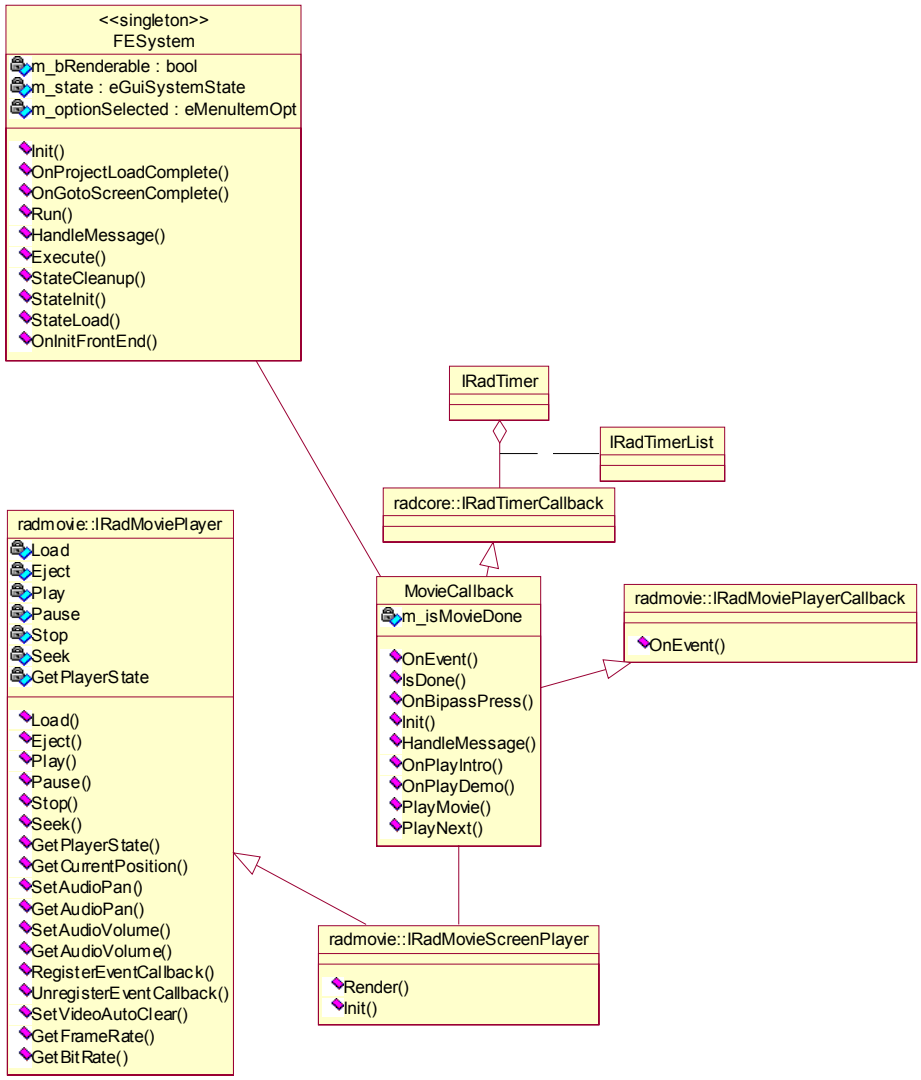

Figure 3 MovieCallback Overview

This Document contains Confidential Information of Radical Entertainment.

3. CLASS SPECIFICATIONS

3.1 guSingleton

3.1.1 Description

Inherit from this template class (binding your child class) to make your class a singleton.

3.1.2 Public Interfaces

The public interfaces provided by the class Singleton and inherited by the FEUserInputHandler are as follows:

- GetInstance: Provides a means by which to access the Singleton class. Returns the one instance of your Singleton child class or if one has not been created yet, first instantiates an object of your child Singleton class and then returns it. This method hides the constructor from the public and limits the number of instances of the class to one.

3.2 ControllerEventMapper

3.2.1 Description

The FEUserInputHandler is derived from this class.

3.2.2 Public Interfaces

The public interfaces provided by the ControllerEventMapper class and used by the FEUserInputHandler are:

- SetEventMap: set the current event map name for the handler.
- AddController: For the specified controller, add to the input event map array a map based on the current event map name and the specified controller.
- Update: For each controller, gets all buttons' down, pressed, and released events and notifies the FEUserInputHandler.

3.3 FEUserInputHandler

3.3.1 Description

Handles controller input notifications (see Update method of section 3.2) by routing them to the FESystem.

This Document contains Confidential Information of Radical Entertainment.

3.3.2 Public Interfaces

The public interfaces provided by the FEUserInputHandler class are:

- Execute: Called during each simulation. Calls the Update() function inherited from the ControllerEventMapper to poll each of the controllers.

3.4 FESystem

3.4.1 Description

This class manages the GUI Menus, InGamePlayPlay HUDs and FMV movie.

3.4.2 Public Interfaces

The public interfaces provided by the class FESystem are:

- Init: Called during game initialization. Gets a pointer to the Scrooby Application. Create and initialize the MovieCallback and the FEUserInputHandler. Note that this should be called after the initialization of the Controller System.

- StateCleanup: Cleans up anything previously allocated by StateInit and StateLoad in the last game state transition that won't be needed in the new game state.

- StateInit: This method is responsible for allocating memory or initializing structures needed for this class in the new game state.

- StateLoad: This method is responsible for loading any data needed for this class in each new state that it will be executing. Will call OnInitPreGP, OnInitInGamePlay, or OnInitPostGP the first time it is called in the change to the GS_PREGP, GS_GAMEPLAY, and GS_POSTGP states respectively. Will return TLS_LOADING each time it's called until the current project has completed loading (see OnProjectLoadComplete).

- Execute: Called during rendering. Draws all currently visible Scrooby images. If the game is currently playing an FMV movie, calls MovieCallback's IsDone() method to see if current movie is finished.

- HandleMessage: Called by the FEUserInputHandler to handle any controller button actions accordingly (activating a screen, showing a sprite, etc). May route message handling to one of the GUI Project Managers or the MovieCallback.

- OnInitPreGP: moves m_state from UNINITIALIZED to PREGP_LOADING. Then loads up the Scrooby PreGameplay project and the Nintendo FMV.

This Document contains Confidential Information of Radical Entertainment.

- OnInitInGamePlay: Moves m_state from PREGP_ACTIVE to INGAMEPLAY_LOADING then loads up Scrooby InGamePlay project.

- OnInitPostGP: Moves m_state from INGAMEPLAY_ACTIVE to POSTGP_LOADING then loads up Scrooby PostGamePlay project.

- OnProjectLoadComplete: Called by Scrooby App class when scrooby is finished loading any project. Sets the m_state to PREGP_ACTIVE, INGAMEPLAY_ACTIVE, or POSTGP_ACTIVE depending on which project was loaded.

- OnGotoScreenComplete: Called by Scrooby when it finishes activating a screen in a project as requested by this class.

3.5 MovieCallback

3.5.1 Description

This class manages the FMV movie.

3.5.2 Public Interfaces

The public interfaces provided by the class MovieCallback are:

- Init: Calls ::radMovieCreateScreenPlayer to therefore create the movie screen player (IRadMovieScreenPlayer) and then calls its Init() method. Register the MovieCallback object with the movie screen player to notify when the end of the movie is reached.

- HandleMessage: Handles messages, mainly from the FESystem, activated either by an A or Start Button press or time lapse which initiate the playing of a movie. Calls OnBipassPress, OnPlayIntro, OnPlayDemo, etc.

- OnPlayIntro: Plays the Nintendo, Pixar, Disney, THQ, and Radical FMVs. Then calls OnPlayDemo unless the A or Start button is pressed during the Radical FMV.

- OnBipassPress: Stop the current movie. If current m_movie is >= Nintendo and < Radical then calls PlayNext(). Otherwise, (the current m_movie should be demo) the gameState is set to GS_PREGP (or other applicable states as other bipassable movies are defined by game design).

- OnPlayDemo: This is called after 5 seconds if user doesn't hit A or Start button while Radical Screen is displayed; or after 30 seconds if user doesn't hit Start button while the Title Screen is active. This sets the m_movie to the demo movie and then calls PlayMovie.

- PlayMovie: plays the movie specified by m_movie.

This Document contains Confidential Information of Radical Entertainment.

3.6 IRadMovieScreenPlayer

3.6.1 Description

This radmovie class is a specialization of the IRadMoviePlayer class. The MovieCallback contains an instance of this class to control playing of FMVs.

3.6.2 Public Interfaces

The public interfaces provided by the IRadMoviePlayer class that is inherited by IRadMovieScreenPlayer to be called by the MovieCallback class are:

- RegisterEventCallback: Registers MovieCallback object to have it's OnEvent() method triggered when the MoviePlayer is finished playing the movie (whether error or finished playing).
- Load: Loads up a movie file. (if not empty calls Eject first).
- Play: Plays the loaded movie file.
- Pause: Pauses play of current movie.
- Stop: Stops play of current movie.
- Eject: Unloads all resources associated with movie player.

This Document contains Confidential Information of Radical Entertainment.

4. DYNAMIC BEHAVIOR

The dynamics of the GUI system is mainly driven by the instigation of a new game state; users selecting options provided in the GUI screens (or lack of user input as in case of Demo Movie); or the end of any NIS or FMV movie.

4.1 Sequence Diagrams

The game state changes are outlined in the state diagram presented in Figure 4 below.

This Document contains Confidential Information of Radical Entertainment.

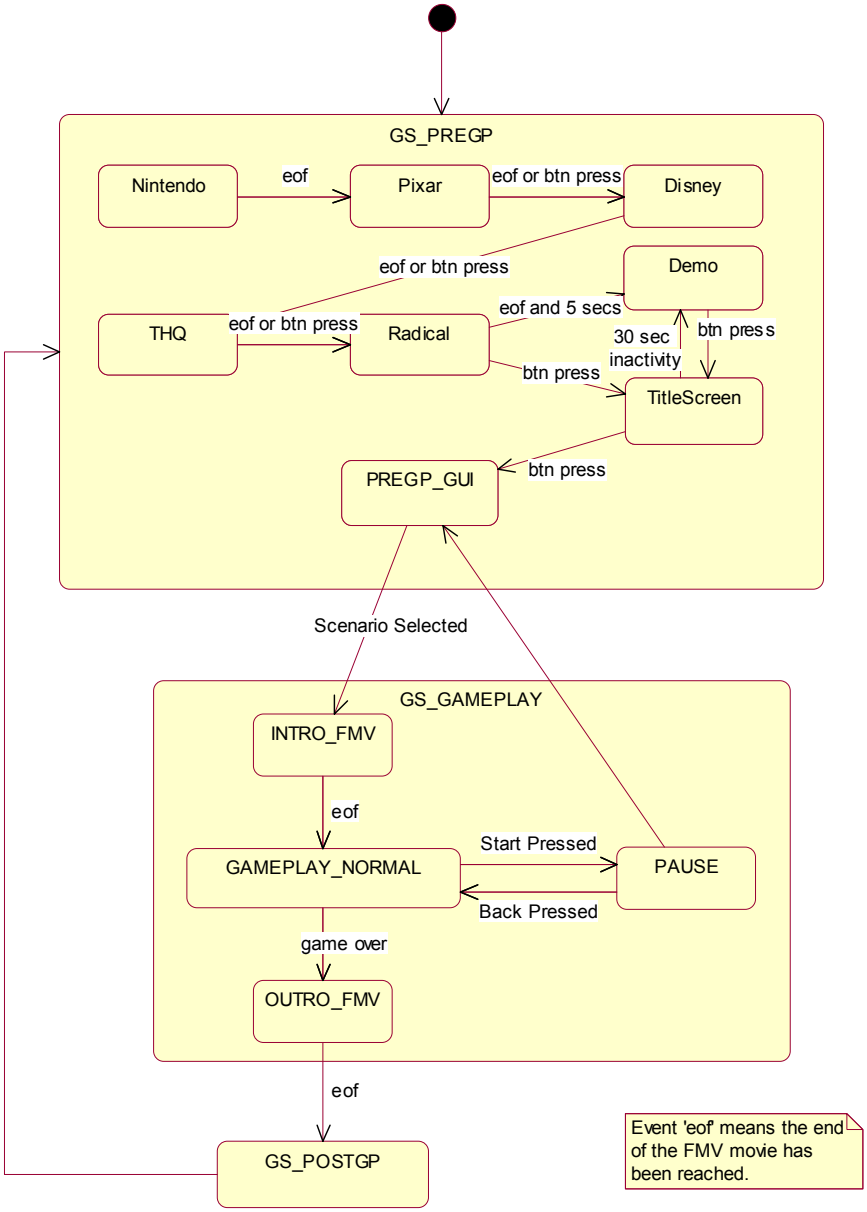

Figure 4 Game State Diagram

During transition to each of the major game states GS_PREGP, GS_GAMEPLAY, and GS_POSTGP the FlowServer calls the FESystem's StateCleanup method to unload and/or delete any obsolete Scrooby Project or Scrooby project GUI Manager. It then calls the StateLoad method to load the appropriate Scrooby project and FMV (if applicable), and then instantiates the corresponding Project GUI Manager. As described in [2], the Flow Server calls the FESystem's StateLoad repeatedly until all of the resources necessary for the FESystem to work

This Document contains Confidential Information of Radical Entertainment.

in the new state are finished loading. This is illustrated in the sequence diagram describing the FESystem StateLoad() calls during the transition to the GS_PREGP state.

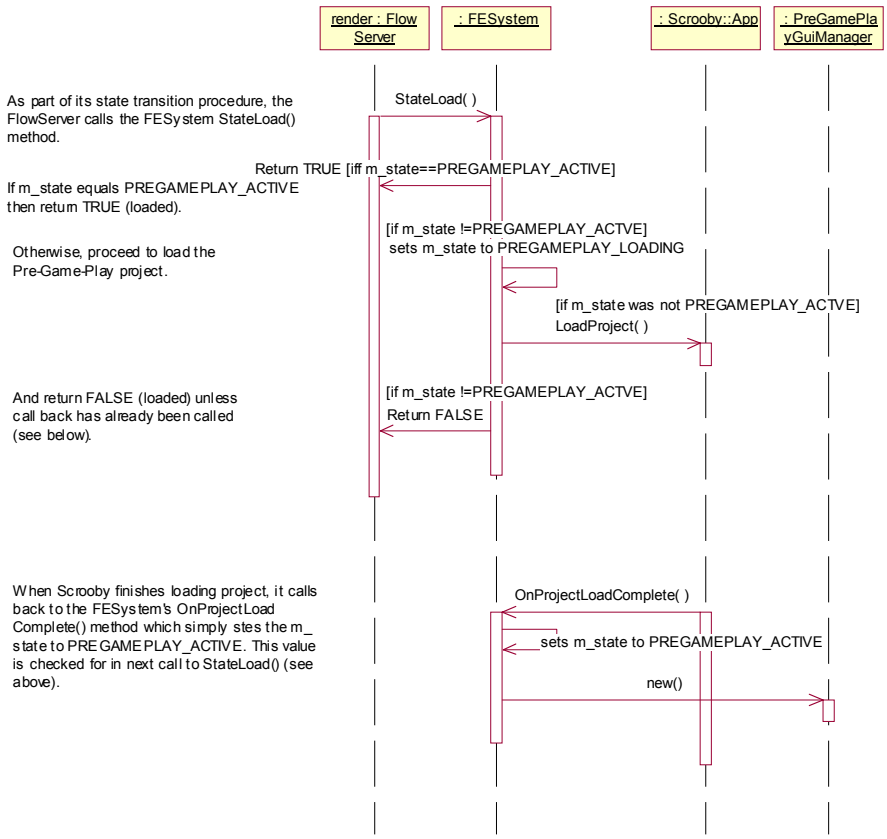

Figure 5 Loading up for the Pre-Game-play game state.

At the beginning of the GS_PREGP state, the GUI System gets the MovieCallback object to start off the first movie (Nintendo). The movie callback takes care of itself in terms of loading and playing each successive movie until demo is finished and it starts the next game state transition to GS_PREGP.

This Document contains Confidential Information of Radical Entertainment.

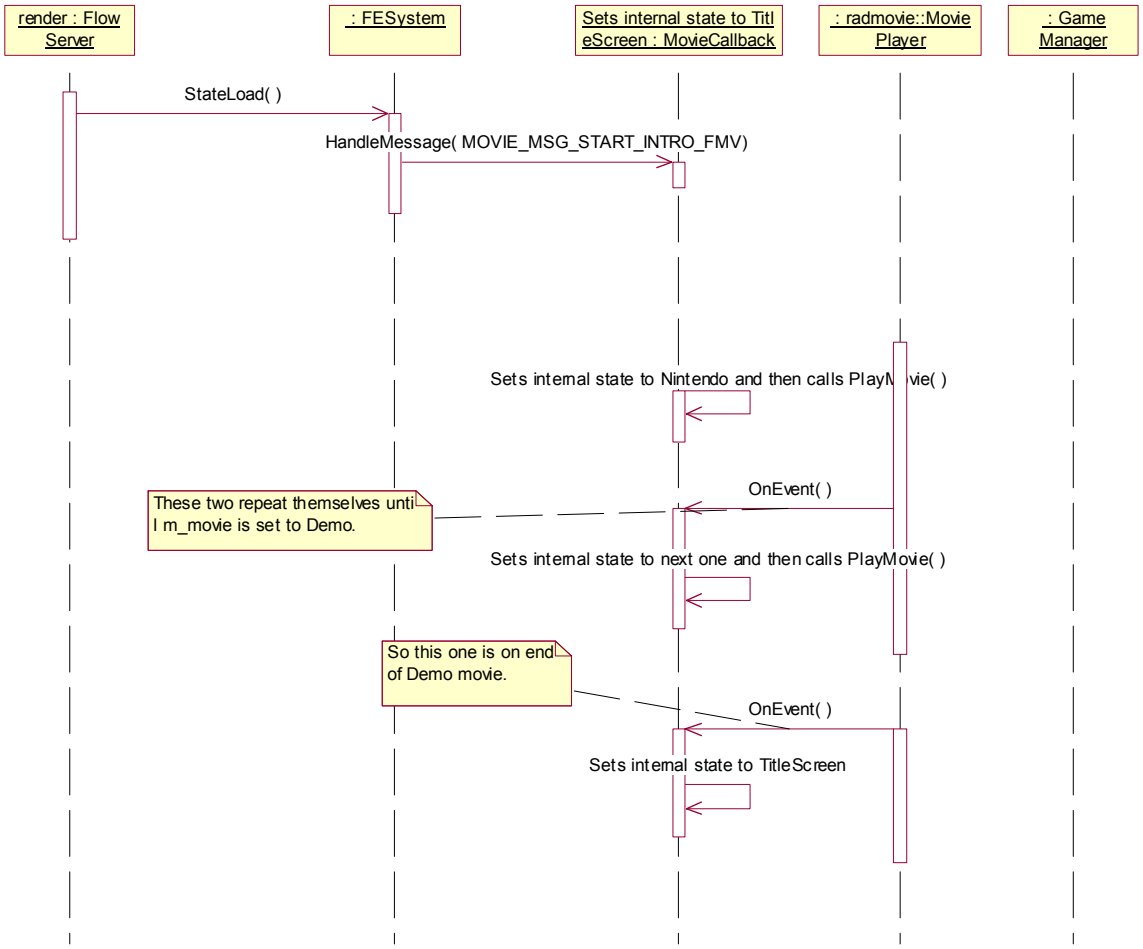

Figure 6 GS_PREGP FMV control

Of course all movies (except for the Nintendo Movie) can be bypassed with the Start or A button. The FEUserInputHandler checks for these action requests during each simulation, then notifies the GUI System to deal with them as shown in Figure 7. When the Radical Movie is playing, an A or Start button bypass will result in a gameState change to GS_PREGP and therefore the display of the Title screen.

This Document contains Confidential Information of Radical Entertainment.

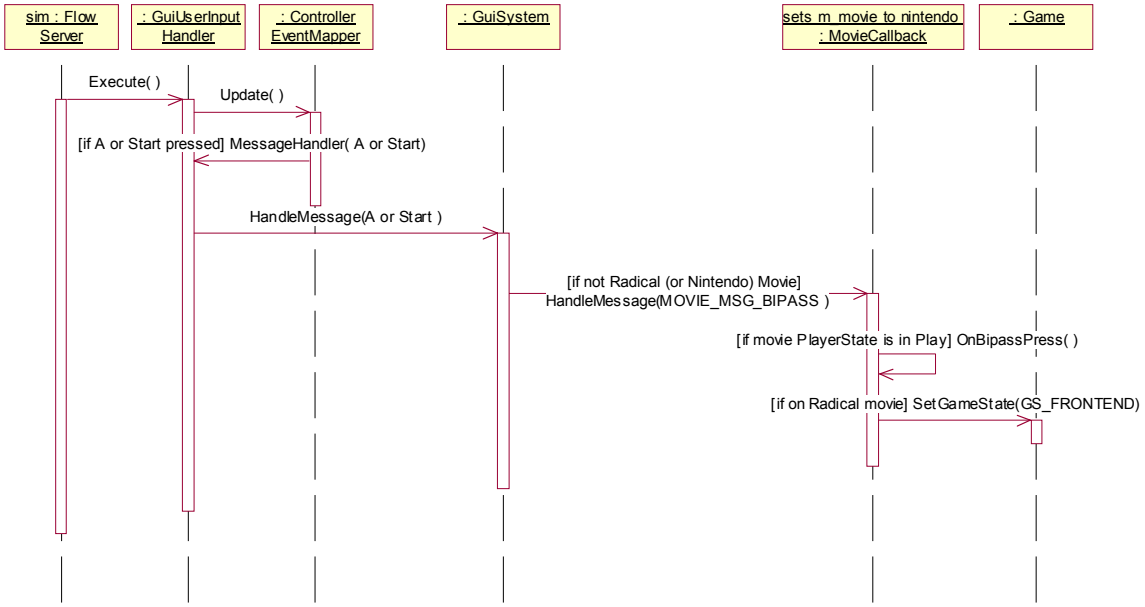

Figure 7 Bipassing movie during GS_INTROFMV gamestate

This Document contains Confidential Information of Radical Entertainment.

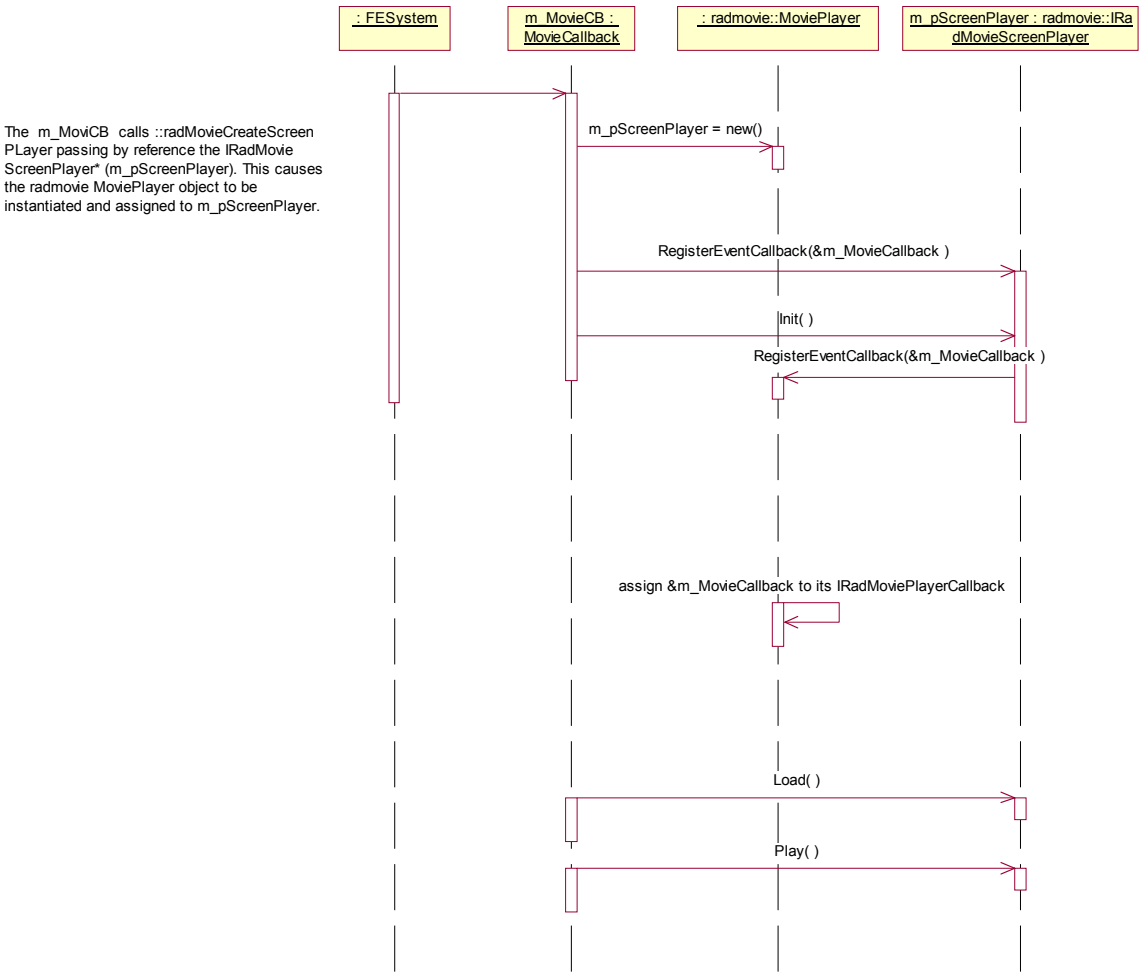

Figure 8 MovieCallback Initialization

This Document contains Confidential Information of Radical Entertainment.

5. PLATFORM CONSIDERATIONS

This game is designed for Nintendo's GameCube platform only.

This Document contains Confidential Information of Radical Entertainment.

6. SUMMARY

6.1 Risks

1. May not be able to implement 'panning' effect desired by artists to implement their really groovy idea for the Pre-Game menus.

2. On the PS2, there was a problem loading a radmovie at the same time a Scrooby Project was loaded. There should not however be a problem with the GameCube.

3. This design requires two Scrooby Projects to be loaded at one time. There is a plan for the Scrooby library to support this soon.

6.2 Dependencies

1. Delivery of Scrooby library update to support the loading of multiple projects.

2. Delivery of GUI and Ingame Hud Art contained within a Scrooby Project provided by the RGB team.

3. Delivery of all necessary FMV movies.

6.3 Assumptions

1. Demo movie is to start play at beginning regardless of where it was when it was last exited.

2. External Company credit FMVs should exist independently (they should not be all one FMV).

3. GUI and InGamePlay art will be contained within the Scrooby Projects as defined above.

4. vxControllerManager::GetNumConnectedControllers always returns a value between 1 and 4 inclusive.

6.4 Overall Effort Estimate

A breakdown of the time it takes to implement each of the classes contained within the Front End is broken down below. Only a small portion of each estimate covers the implementation. The rest of each estimate covers things like unit testing, integration testing, code review updates and design update requests.

Class/Component Name	Estimate in Person Days

This Document contains Confidential Information of Radical Entertainment.

Class FESystem	1.5 Months
Class PreGamePlayGuiManager	1.5 Months
Class InGamePlayGuiManager	1.5 Months
Class PostGamePlayGuiManager	1.5 Months
Class MovieCallback	1 Months
Class FEUserInputHandler	0.5 Months
Total	**7.5 months**

This Document contains Confidential Information of Radical Entertainment.