Autodesk®
Scaleform®

MMO Kit Overview

This document describes the architecture and content of the Scaleform 4.2 MMO Kit, a fully-featured, AAA, reusable user interface solution for a PC MMORPG.

Author: Nate Mitchell, Prasad Silva

Version: 1.00
Last Edited: January 19, 2012

Autodesk®
GAMEWARE

Copyright Notice

Autodesk® Scaleform® 4.2

How to Contact Autodesk Scaleform:

Document	MMO Kit Overview
Address	Autodesk Scaleform Corporation 6305 Ivy Lane, Suite 310 Greenbelt, MD 20770, USA
Website	www.scaleform.com
Email	info@scaleform.com
Direct	(301) 446-3200
Fax	(301) 446-3199

Table of Contents

1 Introduction

The Scaleform MMO (Massively Multiplayer Online Game) Kit, designed for optimum performance and memory savings, provides a foundation for massively multiplayer online game UI using Autodesk® Scaleform®.

The kit includes CLIK-based MMO interface elements like an Inventory and Paper Doll, as well as frameworks for drag and drop, windowing, dynamic icon resource management, and component data binding. The assets, ActionScript widgets, and sample C++ code can be reused by developers and/or leveraged as a best-practices sample in the architecture and implementation of their user interface for their game.

Figure 1: MMO UI Overview

Although this kit provides an out of the box UI solution for an MMORPG, users are not limited by the content provided. We expect users to customize or extend individual elements of this kit to create new and innovative interfaces for any type of game or application.

1.1 Features

The MMO Kit features the following reusable UI widgets:

1. Player and Target Nameplates
2. Spell / Ability Book
3. Paper Doll (Player Equipment and Statistics Manager)
4. Inventory
5. Tooltips
6. Quick-Action Bars
7. Cast Bar
8. Window Menu
9. Chat Log with Tabs

The kit also provides logic for the following frameworks:

1. Drag and Drop
2. Windowing
3. Dynamic External Resource Management
4. Game-Data Binding to Widgets

2 Overview

2.1 File Locations and Build Notes

The files associated with this demo are located in the following locations:

- *Apps/Kits/MMO/* - Contains C++ code for the MMO Kit demo executable.
- *Bin/Data/AS3/Kits/MMO/* - Contains Flash assets and ActionScript code.
- *Projects/Win32 /{Msvc80, Msvc90, or Msvc10}/Kits/MMO/* – Contains the demo projects for Visual Studio 2005/2008/2010 on Windows.

A pre-built executable of the demo for Windows, MMOKit.exe, can be found in the *Bin/Kits/MMO.* It is also accessible via the start menu or the Scaleform SDK Browser.

On Windows, the Scaleform 4.2 Kits.sln file located in the *Projects/Win32/Msvc80/Kits* (or *Msvc90/Kits or Msvc10/Kits*) directory can be used to build and run this demo. Be certain that the "Working directory" for Debugging is set to the *Bin/Data/AS3/Kits/MMO* directory before running the demo from the solution.

2.2 Demo Usage

The demo starts with the Nameplates, Inventory, Action Bars, Window Menu, and Chat Log loaded and displayed on-screen.

2.2.1 Nameplates

Figure 2: Player and Target Nameplates

The Player and Target Nameplates are at the top left of the user interface. The nameplate on the left displays the name, level, portrait, health, and mana (magical spell power) of the Player. The nameplate on the right displays the same information for the Player's current target. The name and level information are bound to game data and will be updated automatically. The health and mana bars are not bound as they are currently not affected in the back-end simulation.

2.2.2 Inventory

Figure 3: Inventory Bar

The Inventory Bar at the bottom right can be used to open and close the Player's inventory. The inventory is divided into separate backpacks, each represented by a toggle button on the bar. When a bag is opened, a new inventory window is shown that displays the contents of the bag.

Each bag can be moved to another slot in the inventory bar. If a bag contains no items, it can be placed within another bag. If a bag is not empty and moved to an occupied slot on the Inventory Bar, the bags will swap slots. The right-most bag is the Backpack and cannot be moved from its slot.

Figure 4: Inventory Bag

Each bag has a predefined number of inventory slots available for items. If multiple bags are open at the same time, the Inventory will automatically reflow the positions of the bags' windows so they are all visible simultaneously.

Items can be moved or dragged around the inventory, within a single bag or between bags. To move an item, left click or start a mouse drag (hold left mouse button down and move the mouse cursor) on the item's icon. During the move, the icon will follow the mouse cursor. If the move was initiated by a left mouse click, left click on the target inventory slot to complete the move. If the move was initiated by a mouse drag, release the left mouse button over the target inventory slot to complete the move.

Certain items can be stacked on top of one another. Slots with stacked items will display a small number at the bottom right hand corner of the slot denoting how many items are in the slot's stack. Examples of items that might be stacked are consumables like potions or food. In the MMO Kit, the

health potions (red bottle icon) and mana potions (blue bottle icon) in the bags can be stacked on top of one another, which will increase the stack size.

If an item is placed in an already occupied inventory slot and it cannot be merged with the other item, the two items will swap inventory slots. Any item, like potions, can be dragged to an Action Bar slot for later use. This creates a direct link between the Action Bar slot and the inventory item. If an Action Bar item is created for a consumable item or stack, then that item will the total number of items of that type in the player's inventory. Equippable items, like a sword or armor, can be dragged into the Paper Doll to be equipped.

2.2.3 Action Bars

Figure 5: Action Bars 1 and 2

Each Action Bar is made up of 12 Action Bar slots. Each slot acts as a drag and drop target (for changing the ability or item in the slot) as well as a button (for using said item or ability). There are three separate Action Bars by default: two horizontal bars at the bottom center of the interface and one vertical bar along the right edge of the screen. Left clicking on an icon in an Action Bar slot will attempt to use that skill or item.

The skills and items in the Action Bar slots can be moved or swapped by dragging an icon into a different slot. Skills or items can be removed from an Action Bar by dragging them off of the Action Bar and completing the move. This does not destroy the skill or item, rather it destroys the link. Skills can be added to the Action Bar by dragging them from the Spell Book into an Action Bar slot. Items can be added to the Action Bar by dragging them from the inventory into an Action Bar slot.

Note that the Action Bar slots support global and individual "cooldowns", time periods during which the ability within cannot be used. Cooldowns are represented by a radial swipe over the slot that progresses from a full tinted square to nothing as it moves clockwise starting from the top center of the slot.

2.2.4 Cast Bar

Figure 6: Cast Bar

The Cast Bar shows the user the name of the skill being used and the remaining “cast time”, the time until the action is actually performed, as both a number and as a progress bar. It is displayed when the player uses an ability which will not be immediately performed. When the cast time is complete and the action is performed, the Cast Bar will fade out and remain hidden until the next action is performed.

2.2.5 Tooltips

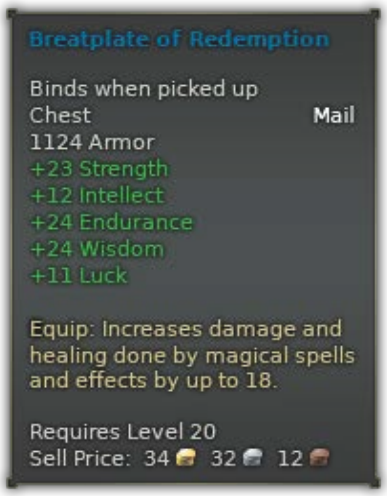

Figure 7: Tooltip

In most MMORPGs, all items and abilities have data associated with them that is relevant to the user but requires more real estate to display than a small icon can provide. For example, a sword item that the player can equip may have internal data about its speed, damage, and weight that the player needs to determine if they can use it. A common method of presenting this information to the user is to display a tooltip when the user mouses over the item. When the user’s mouse leaves the item/ability’s space, the tooltip will be hidden.

The MMO Kit’s tooltip is populated with data from the backend and is customized on a per item/ability basis. Tooltips primarily present colored text and images via HTML, but can be further customized based on the needs of the game. By default, the tooltip will be displayed at the top left of the icon; however, if the tooltip is set to be displayed partially off-screen due to the location of the icon, its position will be adjusted to ensure it is displayed in visible space.

2.2.6 Window Menu

Figure 8: Window Menu

The Window Menu at the bottom left is designed to provide users a way to toggle the various widgets that are not always on-screen (e.g. Paper Doll, Spell / Ability Book, etc…) without using keyboard shortcuts. The Window Menu is made up of a set of toggle buttons, each with a unique icon to represent the screen that the button opens.

In the MMO Kit, the Window Menu can be used to open and close the Paper Doll and Spell Book. Although the other buttons are currently disabled, they can be easily enabled and hooked up to toggle new widgets created by the user.

2.2.7 Paper Doll (Player Equipment and Statistics Manager)

Figure 9: Paper Doll

The Paper Doll, or Equipment Manager, shows the player detailed information about his character and his currently equipped weapons and armor. It allows the player to equip or remove items from his inventory.

The Paper Doll is made up of three sections: a ScrollingList on the left side allows you to toggle the information displayed in the center panel; an equipment manager with a set of Paper Doll's slots for managing equipped items (this panel is displayed by default, although this panel would change depending on the user's selection on the left); and a ScrollingList on the right side that displays a series of boxes that contain HTML TextFields for displaying pertinent data about the player's character. Note that the player statistics are currently not bound to the data in the backend.

The Paper Doll's slots also allow the developer to limit the subset of items that may be equipped. For example, because there is only one "Head" slot for the player, the "Helm of Redemption", a "Head" slot type item, is the only equipment that the player may equip in that slot. Should the user try to equip a sword in the helm slot, no change will occur and the sword will be returned to its previous slot.

The Paper Doll's slots also support smart swapping functionality. For example, if the player has a one-handed sword equipped in his left hand and shield equipped in his right hand and then equips a two handed sword, the one-handed sword and shield will be placed in the user's inventory and the two-handed sword will take up the left and the right hand slots. This functionality can be further extended to meet the needs of the game.

2.2.8 Spell / Ability Book

Figure 10: Spell / Ability Book

The Spell / Ability Book is a list of spells and abilities available to the player. These abilities can be dragged from the Spell Book on to the Action Bars for convenient use. Clicking on ability in the Spell Book will cause said ability to be used.

The abilities are grouped by type using a ScrollingList on the left side of the window. Selecting a group will open a new page of abilities and each group may have multiple pages. Each ability has an icon, a name, and a rank which are populated by the backend data. Depending on the player's level, certain abilities may not be available and therefore are disabled from dragging or use.

2.2.9 Chat Log

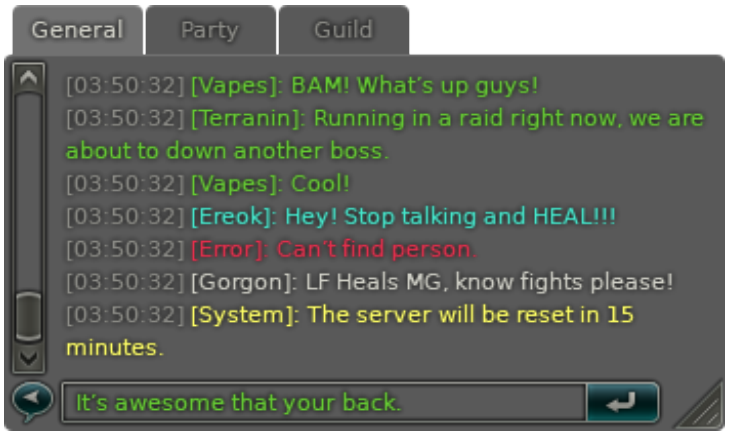

Figure 11: Chat Log

The MMO Kit's Chat Log is a fully functional interface and log for player communication, populated by server data. The log supports multiple chat channels which can be divided and organized into separate chat tabs. The Chat Log's input field also supports sending text to particular channels by using the following prefixes: /s (Say), /1 (World), /p (Party), /g (Guild). Messages targeted to other players can be send via the following syntax: /t <player name> <message>.

The Chat Log displays a history of all chat messages. The length and size of the history is determined by the application.

3 Architecture

The MMO Kit consists of the Flash UI assets and the C++ code that communicates the server's data using an event model that ultimately drives the client view. To demonstrate the integration of the MMO Kit with an existing client-server backend, the kit is built upon a simplistic client-server implementation where server-side simulation generates data and sends updates to the client when appropriate.

3.1 C++

The two main parts of the C++ code are:

1. The environment (server) that provides data to the client.
2. The client's communication interface with the Flash content.

In the MVC paradigm, the communication interface between the client and the content is the controller, the client's data (initially provided by the server) is the model and the Flash elements and ActionScript widgets make up the view.

The code detailed within this document presents one optimal implementation of Scaleform, leveraging the Scaleform Direct Access API to accomplish UI updates driven entirely from native code. However, unlike the HUD and Menu Kits, the MMO Kit combines basic Direct Access API communication with a data binding framework that binds unique string representing backend data to ActionScript UIComponents. Should backend data change, the UIComponents bound to specific identifiers are automatically notified of the data change immediately and can update their views appropriately.

3.1.1 C++ Files

The C++ code consists of the following files (located in *Apps/Kits/MMO/* and its subfolders).

3.1.1.1 Demo

- **MMOKitDemo.cpp** – Core application built on top of the standard Scaleform Player. Handles platform application implementation and initialization of the Game.
- **Game.cpp –** Handles the initialization of and the communication between the Client and Server.

3.1.1.2 C++ and ActionScript Communication Interface

- **GameUIDataBindings.cpp** – Majority of the definitions for the GameUIAdapter class. This class is responsible for the creation of binding slots for the UIComponents in the Flash content. It handles majority of runtime communication between C++ and Flash content. Also contains much of the Slot (Inventory Slot, ActionBar Slot, Equipement Slot, etc…) processing for updating, dragging, dropping, and swapping.
- **GameUIAdapter.cpp** – Core of the GameUIAdapter class. High level function definitions for registering slots, communicating with the client, and updating the cast bar.

3.1.1.3 Client and Server

- **GameClient.cpp** – Definitions for GameClient class. GameClient receives events from the server and processes them by interfacing with the GameUIAdapter which causes the server's changes to be reflected in the UI.
- **GameServer.cpp** – A simple server implementation for the MMORPG. Simulates an MMO server and provides the GameClient with data to display in the UI. Handles messages from the GameClient based on the user's interaction with the Flash UI.

3.1.1.4 Runtime Atlasing

- **GameUIRuntimeAtlasing.cpp –** Provides sample logic that packs a set of external images into a texture atlas as to reduce the overall number of draw calls to the GPU. For the MMO Kit, all of the individual, external images that make up the icons for items, spells, and abilities, are packed into a single texture atlas at runtime. This allows the artist to easily add new icons without creating and managing texture atlases.

3.2 Flash

The Flash content for the MMO Kit can be found in the *Bin/Data/AS3/Kits/MMO/* directory. All directory paths provided in this section will be relative to this directory. A FlashDevelop project is provided, MMOKit.as3proj, which can be used for more convenient navigation and modification of the associated ActionScript codebase.

The widgets that make up the UI (eg. Paper Doll, Inventory, Action Bars) are separated into individual FLA files. Each FLA file provides the graphical layout, images, and animations for a widget. All of the ActionScript logic that makes the widget FLAs interactive is located in the *com/scaleform/mmo* directory. Each widget has its own ActionScript classes that are divided into sub-folders based on the associated widget.

For example, the Flash content for the Inventory widget can be found in *widgets/**InventoryView.fla**.* The AS for the Inventory is located in *com/scaleform/mmo/inventory/*, where InventoryView.as is the Document Class*.*

3.2.1 MMOKit.fla

The primary FLA for the kit can be found in the root of this directory: MMOKit.fla. This is the first file loaded by the MMOKitDemo application at runtime. MMOKit.as serves as the entry point for all ActionScript logic in the kit and is primarily responsible for loading all widgets and initializing the ActionScript frameworks like drag and drop.

All of the ActionScript for MMOKit.fla is external to the FLA. The primary AS file is the Document Class for the FLA, com.scaleform.mmo.MMOKit, located in the *com/scaleform/mmo/* directory.

The MMOKit class takes care of initialization of the following frameworks and subsystems during its configuration (configUI()), which occurs immediately following the first frame (Event.ENTER_FRAME):

- TooltipManager
- WindowManager
- DragManager
- DataBinding
- GameDataModel

The configUI() method also configures the custom Mouse cursor and loads initial widgets: the Chat Log, Window Menu, Inventory, Action Bars, Nameplates, and Cast Bar.

3.2.2 Drag and Drop Framework

The MMO Kit includes a CLIK-based drag and drop framework built on two primary classes: DragManager and DragSlot.

DragSlot is the Button-based class that acts as a Button that can hold data which may or may not include an icon. The Button itself can be used to interact with the data within or moved about the UI to other DragSlots. The DragSlot is responsible for initializing the initial drag event upon user interaction. The DragManager handles the attachment of the DragSlot's data/icon to mouse movement and communication between the original DragSlot, where the drag began, and target DragSlot, where the drag ends. Note that the DragManager must be initialized for the DragSlot component to work.

The DragSlot class is the base functionality and skeleton for all interactive slot elements in the UI. DragSlot should be extended to properly interface the events with the game's backend. Many important functions, including startDrag() and endDrag(), are stubs. These stubs are designed to be

overridden by subclasses implemented by the developer. This allows for DragSlot to be the base class for many different type of slots throughout the UI.

For example, a developer may want a drag that starts in an ActionBarSlot and ends in an InventorySlot to do nothing; a drag that starts in an InventorySlot and ends in an ActionBarSlot to create a link to the inventory item; and a drag that starts in an InventorySlot and ends in an InventorySlot to move or swap the item. By subclassing DragSlot, users can create unique behaviors for each type of slot in the UI.

In the case of MMO Kit, each of the Widgets (Inventory, Spell Book, Action Bars, etc…) has its own implementation of DragSlot. Note that all of these individual classes extend com.scaleform.mmo.core.MDragSlot, a base class which extends DragSlot, which includes basic logic for registering and unregistering the MDragSlot with the GameUIAdapter in the backend..

3.2.3 Windowing Framework

The MMO Kit includes a basic CLIK-based windowing framework that supports opening, closing, moving, and swapping focus to windows within the UI. The core of this framework is the Window Manager class, which provides functions for the opening a new window, changing focus, and swapping window depth based on the last window clicked by the mouse.

The windowing framework uses the com.scaleform.mmo.core.MWindow class for the logic that makes the windows interactive. The MWindow symbol in MMOKit.fla is used for the visual shell of the basic window: the title bar, and the buttons that allow for resizing and closing the window. The Paper Doll and Spell Book widgets use MWindow class to embed the widget within the context of a UI window.

The MWindow Symbol uses Scale9Grid to embed content of arbitrary sizes without distorting the window's edges and corners. The background of the Window is a medium sized bitmap which is scaled up or down depending on the size of the content within. Minor artifacting of the background may occur if the window content is much larger than the original background bitmap's size, however it is only noticeable upon close inspection. For this reason, it may make sense for developers to have background bitmaps of multiple sizes that can be used based on the size of the content in the Window.