

# Kingdom Reborn: Custom UI

## Changing the Paperdoll

## Introduction

This document is a walkthrough on the process I took to create a custom UO: Kingdom Reborn interface. By following this document, you should gain an basic understanding on the inner workings of custom interfaces for UO. This walkthrough is geared strongly towards the XML portion of the interface creation, not on creating appropriate textures to use, so look elsewhere if you just need help in that area. If you want to learn the XML, however, this guide should be of use. One thing this guide won't do, however, is help you read XML. XML is fairly easy to read, but if you have no experience programming, scripting, or writing HTML, you may find it difficult to follow.

One thing I will mention though about XML syntax is how you add comments to the XML. A comment in XML is part of the file that is there to hide some of the text from a program that is reading it. This text is usually used for adding descriptions for humans reading the file, but it sometimes will also contain no-longer-used, but still important, XML. To write a comment in XML you must surround your commented text by “<!--” and “-->”. For example, if you see a chunk of text in an XML file that says something like “<!-- Menu buttons -->”, that is there to tell a human reader that the next area in the XML defines some menu buttons.

## Getting Started

Before you can start with creating a custom interface for UO, there are a few steps you'll need to take.

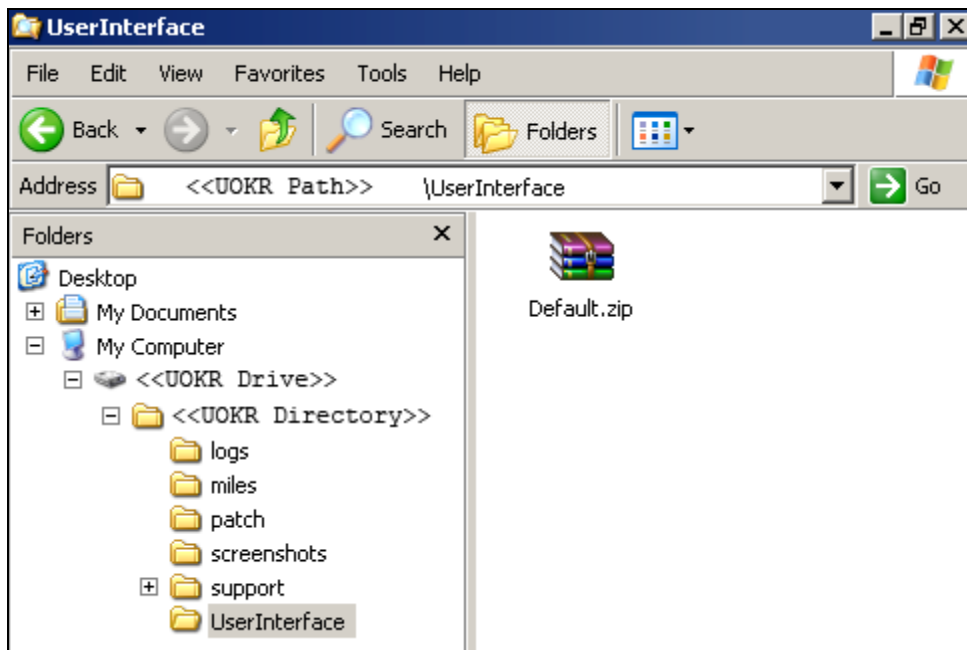
1. Decide what kind of interface you want, and give it a name.

This is sometimes the hardest step – deciding what to make. For this walkthrough, I'm going to be adjusting the Paperdoll to make it more like a “classic” UO Paperdoll, and removing the Menu buttons from the bottom of the screen, since those will now be on the Paperdoll.

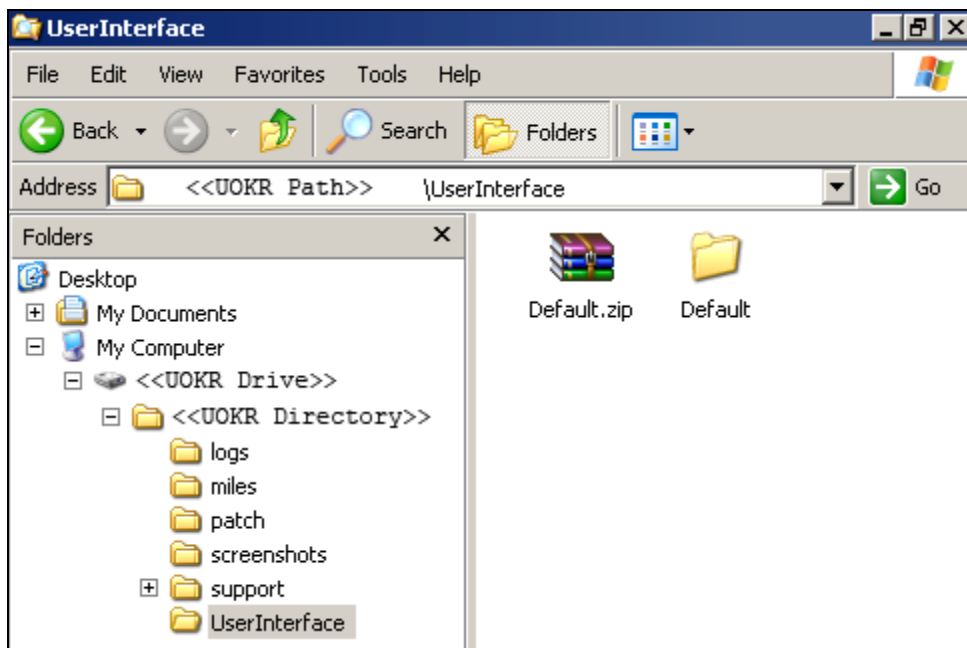
The final name for your interface doesn't need to be decided now, and can be changed very easily, but you will need some name before you can start building. We're going to call the interface we're working on in this walkthrough “Classic Paperdoll”.

2. Extract the default interface files.
3. Navigate to your Kingdom Reborn user interface folder. It is named `UserInterface`, and is located in directory Kingdom Reborn is installed to.

In this directory, you should find a file named `Default.zip`.



4. Extract Default.zip to the UserInterface folder.

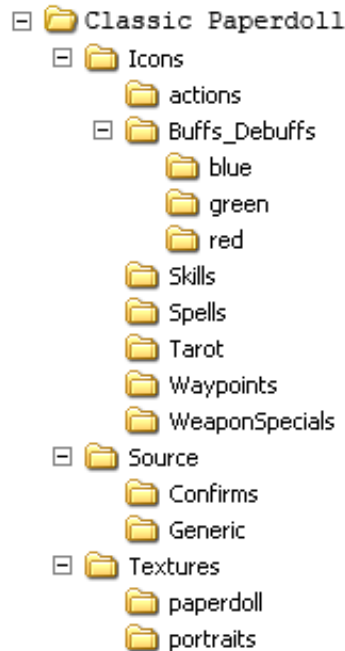


A new folder within your UserInterface folder called Default now exists. In this folder are the source layouts and artwork. You should not change the contents of these files. Instead, you will be making copies of these files in your custom UI's directory, and making changes to the copy.

5. Create the directory structure for your interface.
6. Create a folder with the name of your interface in the UserInterface folder.  
For our interface, we'll create a folder called Classic Paperdoll.
7. (Optional) Inside this folder, create new folders so that the directory structure matches that of the Default UI.

You don't need to create these folders now, and won't need all of them, but I find it useful to just create them all up front and only worry about files later. If you want, you could copy all the files from the Default UI folder into your new folder, but then you'll have extra files in your directory and things might start to get confusing.

You should now have a directory structure that looks like this:



8. Start working.

## Learning the Parts

Before we can make changes, we need to understand what it is we are changing. Looking through the Default UI folder, you'll find lots of XML files in the Source directory. The two that we're going to be most interested in are `menubarwindow.xml` and `PaperdollWindow.xml`. You can open these files with any text editor, or a special XML editor if you prefer. Notepad works just fine though.

We'll start with the `menubarwindow`, since that one is the simpler of the two. It is still quite complicated though, so I'm just going to touch on the important parts.

### *Menu Bar XML File*

The first important things to notice are which script files are being used.

```
1 <Interface xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../../Interface.xsd">
2   <Scripts>
3     <Script file="Source/MenuBarWindow.lua" />
4   </Scripts>
5 </Windows>
```

This tells you where any event handler functions reside, and you'll need to shift the appropriate script files to your XML so that they work properly. There are actually a few other script files being used that aren't listed here, so not all scripts you see will be using this `MenuBarWindow.lua` script file. If the function starts with "MenuBarWindow", that function uses the `MenuBarWindow.lua` script file, and

you'll need to use that script file in whatever XML file you want to use those functions in.

The next important bit is one of the templates for this file – the MenuButtonTemplate.

```
5 <Windows>
6 <Button name="MenuButtonTemplate" sticky="false"
  backgroundtexture="hud_image" highlighttexture="hud_image">
7   <Size>
8     <AbsPoint x="48" y="48" /> (c)
9   </Size>
10  <EventHandlers>
11    <EventHandler event="OnLButtonUp"
  function="MenuBarWindow.ToggleMenuButton" /> (d)
12    <EventHandler event="OnMouseOver"
  function="MenuBarWindow.OnMouseoverMenuBtn" /> (e)
13  </EventHandlers>
14 </Button>
15
16 <Window name="MenuBarWindowBackground" movable="false" layer="background">
```

Here we see a few common definitions:

- a. Where the standard textures are defined (more on textures in a little bit)
- b. Where the highlight textures are defined
- c. The size of the Menu Button
- d. What gets performed when the left mouse button is released on the Menu Button
- e. What gets performed when the mouse cursor is hovering on top of the Menu Button.

Without having the script files, it's hard to know exactly what most event handlers do, but if they exist in the Default UI, you probably want to keep them in your UI. Some of the event handlers are quite a bit more detailed in their naming though, and do actually tell us what they do.

Continuing on in the file...

```
14 </Button>
15
16 <Window name="MenuBarWindowBackground" movable="false" layer="background">
17   <Size>
18     <AbsPoint x="10" y="66" />
19   </Size>
20   <Anchors>
21     <Anchor point="bottomleft" relativePoint="bottomleft" />
22     <Anchor point="bottomright" relativePoint="bottomright" />
23   </Anchors>
24   <Windows>
25     <!-- Menu bar background -->
26     <HorizontalResizeImage name="$parentGoldLeftEdge" texture="hud_image"
  handleinput="false" layer="background" sticky="false">
```

Here we see the start of the section that details the background for the menu bar. This isn't on our list of things to work on just yet, so we'll skip over this for now.

Next in line is the war/peace button definition.

```

74 </Window>
75
76 <!-- War button -->
77 <Button name="MenuBarWarButton" movable="false"
backgroundtexture="hud_image" highlighttexture="hud_image">
78 <Size>
79 <AbsPoint x="61" y="63" />
80 </Size>
81 <Anchors>
82 <Anchor point="bottomleft" relativePoint="bottomleft">
83 <AbsPoint x="10" y="0"/>
84 </Anchor>
85 </Anchors>
86 <EventHandlers>
87 <EventHandler event="OnLButtonUp"
function="MenuBarWindow.ToggleWarModeButton" />
88 <EventHandler event="OnMouseOver"
function="MenuBarWindow.OnMouseoverMenuBtn" />
89 </EventHandlers>
90 <TexCoords>
91 *** <Normal x="208" y="390" />
92 *** <NormalHighlit x="361" y="390" />
93 *** <Pressed x="296" y="390" />
94 *** <PressedHighlit x="449" y="390" />
95 </TexCoords>
96 </Button>
97
98 <Window name="MenuBarWindowStatusBar" movable="false" layer="default">

```

The things that are most interesting to us here are the event handlers, since we want to add the functionality of this button to our paperdoll. The OnMouseOver event seems like it might not be something we need to have, but the OnLButtonUp function sounds like it causes you to enter or leave war mode – definitely something we want on our paperdoll.

Shortly after the event handler, you'll see something we haven't touched on yet, but something we'll need to be aware of when we start making changes on the Paperdoll itself – texture coordinates (TexCoords). Texture coordinates are defined different for each type of element – in this section, we see how it's done for Buttons. There are five different textures for buttons (there may be others, actually, but the most I have seen is five).

1. Normal – This is the texture the button usually displays, when the button isn't active, and the mouse isn't hovering over it. The texture for this is located in the texture defined by backgroundtexture.
2. NormalHighlit – This is the texture that displays on top of the normal texture when somebody holds their mouse over the button. The texture for this is located in the texture defined by highlighttexture.
3. Pressed – This is the texture the button displays while it is being pressed. The texture for this is located in the texture defined by backgroundtexture.
4. PressedHighlit – You guessed it, the texture that is displayed on top of the pressed texture when somebody holds their mouse over the button while it is pressed. The texture for this is located in the texture defined by highlighttexture.
5. Disabled – This is the texture that is displayed if the button is in a state where the user isn't allowed to press it. It is fairly rare for a button to be disabled in UO:KR, so you'll actually see

that this texture isn't even defined for many buttons. The texture for this is located in the texture defined by backgroundtexture.

The coordinates for each of these textures is the pixel-coordinate for the top-left corner of the texture. The size of the texture is defined above, in the Size tag (61x63 in this case).

Next in the file are the Menu Buttons. Here you'll see the code for each of the buttons, as well as two commented out buttons (a Help button and a Macros button). We'll just look at one of the uncommented ones in a little more detail, since they're all pretty much the same.

```
179     <Button name="$parentToggleMap" inherits="MenuButtonTemplate">
180         <Anchors>
181             <Anchor point="topleft" relativePoint="topright"
relativeTo="$parentToggleMainMenu"/> ***
182         </Anchors>
183         <TexCoords>
184             <Normal x="0" y="370" />
185             <NormalHighlit x="145" y="370" />
186             <Pressed x="96" y="370" />
187             <PressedHighlit x="145" y="370" />
188             <Disabled x="0" y="370" />
189         </TexCoords>
190         <EventHandlers>
191             <EventHandler event="OnLButtonUp"
function="MenuBarWindow.ToggleMapWindow" />
192             <EventHandler event="OnMouseOver"
function="MenuBarWindow.OnMouseoverMenuBtn" />
193         </EventHandlers>
194     </Button>
```

The big thing to look at once again is the event handlers. There are what we'll want to be pulling over to the paperdoll. If you look closely though, you'll see that the Main Menu and Friends List buttons don't have event handlers defined for them. We may need to look at those a little bit more when we get to using them.

Another thing to take a look at is the Anchor tag. Anchor tags define the positioning of elements in the UI with respect to other parts. This tag says that the Map button (the button this section is defining) has it's top right corner positioned at the same point as the Main Menu button's top left corner. Basically, this button is located immediately to the left of the Main Menu button. We'll see in other sections that you can say that the button should be a few pixels away, instead of touching, other elements (which is probably what we'll want to do for our paperdoll's buttons).

If we look at the two commented buttons, we can see that they have event handlers listed, so they might work. We'll keep those in mind later to test out, since they might be useful to add to the Paperdoll. It is possible though that the scripts for those buttons have been commented out as well, or that the scripts for those buttons simply weren't finished, so we might find that they won't work for us.

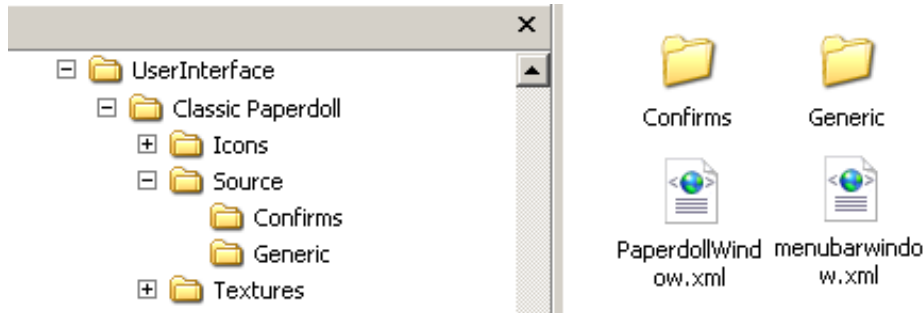
## ***Paperdoll XML File***

Let's move on to the PaperdollWindow.xml file. This file is rather large, so we'll just touch on a few points. You'll see the various parts of the paperdoll as you go through the file: Tabs, Item Slots, the window itself, three tab windows (armor, clothing, profile), the Menu buttons (character sheet and backpack), and the title of the window. We'll want to pay close attention to the first two tab windows (armor and clothing), and the item slots within those, the main window, and the Menu buttons.

\*\*\* More details \*\*\*

## Adjusting the Paperdoll

Before we start any work, we should start by moving the two XML files we've already talked about (PaperdollWindow.xml and menubarwindow.xml) into our custom UI's Source directory.



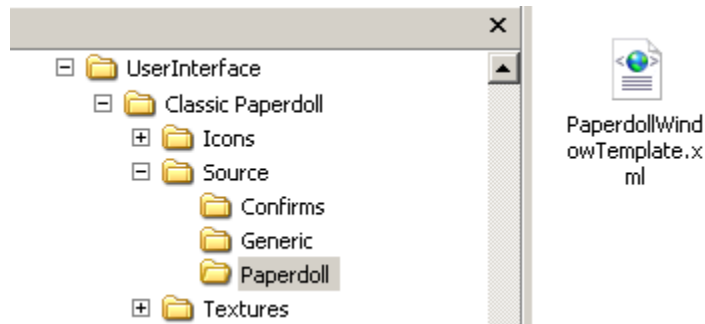
Now let's open up the PaperdollWindow.xml file. The first thing we'll want to do is get our background looking right. To do this, we'll want to scroll until we find that part of the XML file.

```
76 <Window name="PaperdollWindow" movable="true" >
77 <Size>
78 <AbsPoint x="366" y="630" />
79 </Size>
80 <Anchors>
81 <Anchor point="topleft" relativePoint="topleft">
82 <AbsPoint x="1025" y="0" />
83 </Anchor>
84 </Anchors>
85 <EventHandlers>
86 <EventHandler event="OnInitialize"
function="PaperdollWindow.Initialize" />
87 <EventHandler event="OnShutdown" function="PaperdollWindow.Shutdown" />
88 <EventHandler event="OnUpdate" function="PaperdollWindow.OnUpdate" />
89 </EventHandlers>
90 <Windows>
91 <!-- Default chrome. Alter this at your own peril. -->
92 <Window name="$parentChrome" inherits="UO_DefaultWindow">
93 <Anchors>
94 <Anchor point="topleft" relativePoint="topleft"/>
95 <Anchor point="bottomright" relativePoint="bottomright"/>
96 </Anchors>
97 </Window>
98 <!-- End of default chrome boilerplate code -->
99
100 <Button name="$parentTabButton1" inherits="PaperDollTabButton" id="1">
```

One problem with this, however, is that there's no texture detail in our file for the Paperdoll window background. That section that is marked as "Default chrome. Alter this at your own peril..." well, I don't know about the peril part, but this is what we want changed. We can see that it inherits from "UO\_DefaultWindow", so we'll have to find that template to figure out what we need to adjust. The template is located in a file called "UO\_DefaultWindow.xml", so we need to head over to the Default UI's Source folder and open up that file.

As you can see, the definition for the DefaultWindow template is rather large, and we don't want to mess with it directly or else we'll be making almost every window look like our paperdoll window. Instead, what we're going to want to do is actually create a new file for our paperdoll's background, and use that instead.

To help keep things organized, let's create a new directory in our custom UI's Source folder named "Paperdoll", and make a copy of the UO\_DefaultWindow.xml file in that folder, but rename it "PaperdollWindowTemplate.xml".



Open up our new PaperdollWindowTemplate.xml file, and make a change to the name of the window in it to Illandril\_PaperdollWindow:

```
6 <Windows>
7   <Window name="Illandril_PaperdollWindow" movable="false" layer="background"
  popable="false">
8     <EventHandlers>
```

We also need to update our PaperdollWindow.xml file to use this. To do this, we'll need to change the inheritance for the window, as well as include our new file so that KR knows where to find this new template.

```
1 <Interface xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../../Interface.xsd">
2   <Include file="Source/Paperdoll/PaperdollWindowTemplate.xml" />
  ...
  ...
92   <!-- Default chrome. Alter this at your own peril. -->
93   <Window name="$parentChrome" inherits="Illandril_PaperdollWindow">
94     <Anchors>
95       <Anchor point="topleft" relativePoint="topleft"/>
96       <Anchor point="bottomright" relativePoint="bottomright"/>
97     </Anchors>
98   </Window>
99   <!-- End of default chrome boilerplate code -->
```

Now we can go back and try and find those textures. In our new PaperdollWindowTemplate.xml file, you'll see that the area commented as "Body Background" inherits from UO\_Default\_Black\_Background – this is where we'll find our texture definition. This template is tucked away in the CoreWindowTemplates.xml file.

```

35     <!-- Default Black Background -->
36     <FullResizeImage name="U0_Default_Black_Background" layer="background"
    handleinput="false" texture="U0_Background">
37         <Sizes>
38             <TopLeft x="61" y="52" />
39             <Middle x="78" y="78" />
40             <BottomRight x="61" y="66" />
41         </Sizes>
42         <TexCoords>
43             <TopLeft x="0" y="0" />
44             <TopCenter x="68" y="0" />
45             <TopRight x="163" y="0" />
46             <MiddleLeft x="0" y="68" />
47             <MiddleCenter x="68" y="68" />
48             <MiddleRight x="163" y="68" />
49             <BottomLeft x="0" y="158" />
50             <BottomCenter x="68" y="158" />
51             <BottomRight x="163" y="158" />
52         </TexCoords>
53     </FullResizeImage>

```

This little section of XML tells us how we're going to format our texture definitions for FullResizeImage elements, and doesn't inherit from anything so we know this is as deep as we'll need to go. Let's head back to our PaperdollWindowTemplate.xml file, pull out the inheritance, and add in the details we need to have this pull our own texture for the Paperdoll. The big thing we'll want to change right away is what texture to use – let's name our texture file "Paperdoll\_Textures" (we'll actually define that in a moment). We'll keep the coordinates all the same for now, since we don't have our texture ready yet. Your PaperdollWindowTemplate.xml file should now contain the following:

```

6 <Windows>
7 <Window name="Illandril_PaperdollWindow" movable="false" layer="background"
  popable="false">
8   <EventHandlers>
9     <EventHandler event="OnInitialize"
function="UO_DefaultWindow.Initialize" />
10    </EventHandlers>
11
12    <Windows>
13      <!-- Body background -->
14      <FullResizeImage name="$parent_UO_DefaultWindowBackground"
layer="background" handleinput="false" texture="Paperdoll_Textures">
15        <Anchors>
16          <Anchor point="topleft" relativePoint="topleft">
17            <AbsPoint x="4" y="28" />
18          </Anchor>
19          <Anchor point="bottomright" relativePoint="bottomright">
20            <AbsPoint x="-4" y="-1" />
21          </Anchor>
22        </Anchors>
23        <Sizes>
24          <TopLeft x="61" y="52" />
25          <Middle x="78" y="78" />
26          <BottomRight x="61" y="66" />
27        </Sizes>
28        <TexCoords>
29          <TopLeft x="0" y="0" />
30          <TopCenter x="68" y="0" />
31          <TopRight x="163" y="0" />
32          <MiddleLeft x="0" y="68" />
33          <MiddleCenter x="68" y="68" />
34          <MiddleRight x="163" y="68" />
35          <BottomLeft x="0" y="158" />
36          <BottomCenter x="68" y="158" />
37          <BottomRight x="163" y="158" />
38        </TexCoords>
39      </FullResizeImage>
40
41      <!-- Title bar -->

```

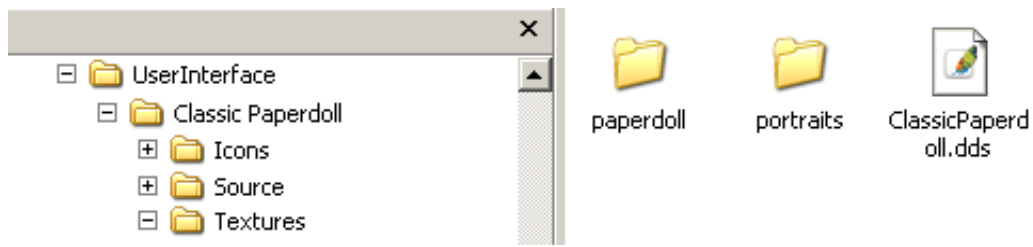
## *Our First Textures*

Before we can continue much farther, we should move over to the textures, and actually get things ready there so we can see some progress in game. Take a look at the Textures.xml file from the Default UI Textures folder. In there you'll see how textures are included.

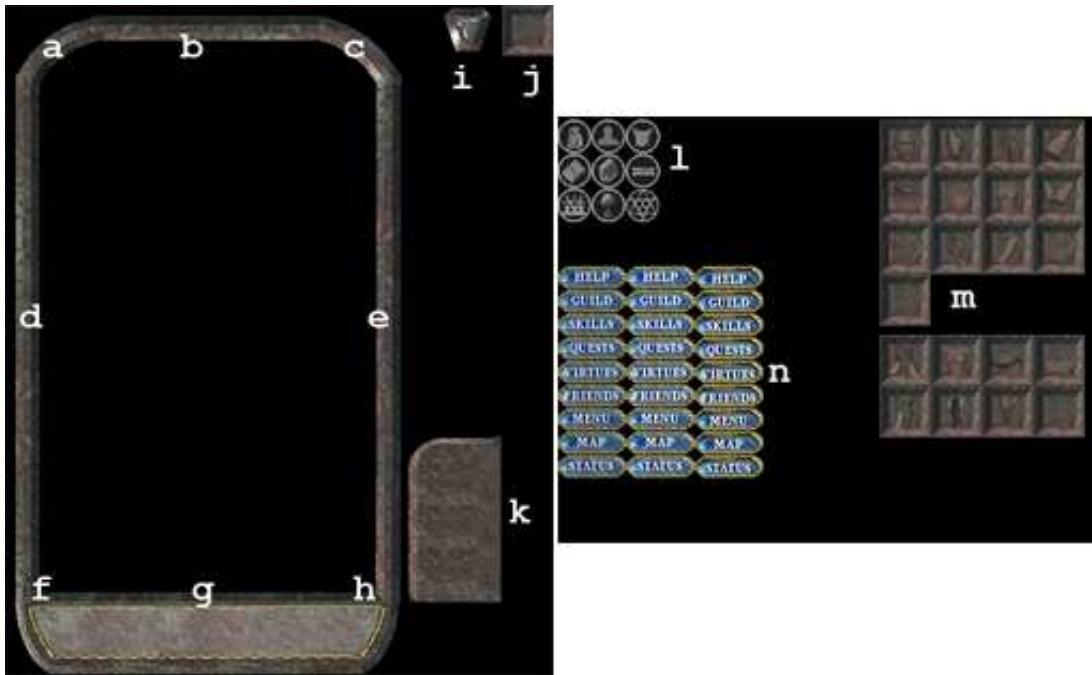
### **\*\*\* Image of Textures.xml \*\*\***

Let's add a texture definition to PaperdollWindow.xml, and bring the ClassicPaperdoll.dds file you got with this walkthrough to our custom UI's Textures folder.

### **\*\*\* Image of PaperdollWindow.xml \*\*\***



Let's take a look at our texture, so we can know what we need to be using for the part we've already got finished.



- a. The top-left corner of the paperdoll window.
- b. The top bar of the paperdoll window.
- c. The top-right corner of the paperdoll window.
- d. The left bar of the paperdoll window.
- e. The right bar of the paperdoll window
- f. The bottom-left corner of the paperdoll window.
- g. The bottom bar of the paperdoll window.
- h. The bottom-right corner of the paperdoll window
- i. The capstone small menu buttons will be placed on.
- j. An empty paperdoll item slot.
- k. A region for large menu items.
- l. Small menu icons.
- m. Paperdoll item slots with indicators on which slot they are for.
- n. Large menu icons.

To use the background for our paperdoll, we need to know which coordinates we want. If we examine the texture, we'll see that the top-left corner starts at the pixel point 10x 10y, so that's what we'll want to use for the top left texture coordinate. The curve of the corner ends 83 pixels to the right, so we'll use 93x 10y for the top center coordinate. The curve of the right corner starts another 200 pixels to the right, so we'll use 293x 10y for the top right coordinate. Going right a little farther, we see that the edge of the paperdoll window is 78 pixels to the right (we'll use this number in a bit). Moving down to the bottom, we can check the x coordinates there as well, to make sure those curves aren't larger. They aren't, so we can use those x coordinates for middle and bottom as well. If they didn't, we'd want to change our top coordinates to fit with the bottom, using the largest width non-repeatable areas to determine our corner sizes.

Working our way from top to bottom, we see that the top-left corner curve stops after 66 pixels, so we'll use 76 as the y value for all our middle coordinates. Going down further, we see that the bottom curves start another 470 pixels down, making 546 the y value we'll want to use for our bottom coordinates. Down another 78 pixels is the bottom of our paperdoll window (we'll use this in just a second).

We now have all of the starting coordinates for our texture segments, but we need to say how big they are before KR can use them. We found that the left section was 83 pixels wide in the widest area; the middle section was 200 pixels; and the right was 78 pixels. The top section was 66 pixels tall in the tallest area; the middle was 470; and the bottom was 78. This gives us 83x66 for the top left, 200x470 for the middle, and 78x78 for the bottom right.

Let's add all these values into our PaperdollWindowTemplate.xml file:

**\*\*\* Image of PaperdollWindowTemplate.xml \*\*\***

Ok... now that we've got this much done, we can see progress! Open up KR, select the skin, and log in.



Too bad most of our progress is being hidden by all those other graphics on top. We'll have to work on that.

## ***Removing Some Excess Clutter***

Let's start our work by getting rid of the frame, the title bar, and the gold plaque looking nameplate.

The first two are quite easy – just open up PaperdollWindowTemplate.xml and delete the lines that define them. We will want to keep the close button, however, at least for now, but we'll want to position it differently. To do that, we'll get rid of the part of the Anchor that says it is relative to the title bar, which will make it relative to the window itself. We can then adjust the positioning so that its top-right is the same as the window's top-right, but we'll keep in the extra AbsPoint details for now in case we want to move it around later. We should also go back up to the window background and change the positioning there to be right at the edges, since we no longer have the frame or title bar to worry about.

**\*\*\* Image of PaperdollWindowTemplate.xml w/ Title Bar, frame, windowtitle, and close button marked for deletion, as well as AbsPoint adjustments\*\*\***

We can now move on to the nameplate. This is located back in the PaperdollWindow.xml file, way down at the bottom.

**\*\*\* Image of PaperdollWindow.xml w/ \$parentBg marked for deletion\*\*\***

Now let's log back in and see what we've got now...



Oh dear! We've lost our names! How did that happen? We kept in the XML that defined the name, didn't we?

Yes, we did, but we removed something else that broke the scripts, unfortunately. When we took out the window title in PaperdollWindowTemplate.xml, we caused the UO\_DefaultWindow.Initialize script to fail (or at least that is what I can gather, mostly just a guess without being able to see the scripts themselves). Without having the scripts to work with, it was pretty much hunt-and-peck to figure out what went wrong, but thankfully we only changed a few things before the last KR restart. It's situations like this that show us why we should keep our changes small until we know they work. If you find something in your UI suddenly disappearing, even though you don't think you changed it, go back and undo the changes you did make one at a time until you get that part back.

So let's go ahead and add the title back in, but adjust its position a little so that it at least fits a little bit –

we'll decide what to actually do with it a little later though.

### \*\*\* Image of PaperdollWindowTemplate.xml.3\*\*\*

And now open up those changes in KR to make sure we have our names back...



Ah, much better. Now let's look at those paperdoll item slots...

### *Tidying Up the Clutter We Keep*

The paperdoll item slots are rather bulky, if we want to keep them as well as add in menu items to the paperdoll. In our new texture, we dropped the size of them down to 48x48 (as opposed to the original 58x58). Let's start by just swapping in our new textures, and making everything smaller.

First, let's hit the ItemSlotButtonDef template. We'll need to shrink down the size of the slots, as well as the texture being used.

### \*\*\* Image of PaperdollWindow.xml.3 showing ItemSlotButtonDef\*\*\*

Next, let's hit each of the buttons. Here we're not only changing the size, but which texture file to look at and where in that texture file. While we're here, we should probably label them all as well.

### \*\*\* Image of PaperdollWindow.xml.3 showing changed buttons\*\*\*

Now let's see those changes look in game...



Nice, tiny, and much more “classic” looking.