****

**Onyx IDE**

**Documentation**

## Contents

### INTRODUCTION

The Onyx Story **3**

Why Choose Onyx? **3**

### THE MENU BAR

File Menu **4**

Edit Menu **5**

View Menu **6**

Help Menu **6**

Themes Submenu **7**

### THE WORK AREA

Class Tree w/UnCodeX

The Text Pane

The Status Bar

### ADVANCED FEATURES

Find and Replace

Tabs and Termination

Autocomplete

The Status Bar

### RANDOM OTHER STUFF

Contact Me

Boring Legal Stuff

The End

# INTRODUCTION

## The Onyx Story

I’ve been working with UDK ever since it came out in 2009. Although I’ve messed around with nearly every part of the engine, my specialty has always been programming: before learning UnrealScript I had taught myself everything from TI-BASIC to C/C++ to ActionScript.

One of the things I noticed almost right away was the lack of an “industry standard” IDE for UnrealScript. For C/C++ I always work in Visual C++ 2010 Express. For Java I either use NetBeans (at home) or Eclipse (at school). And for ActionScript there’s really only one choice: Adobe Flash CS5. Of the hundreds of available IDE’s, I use these for a reason. They satisfy all my needs, are exceedingly easy to work with, and have tons of extra features.

But UnrealScript? I spent the first few months coding out of Notepad. From there I hopped from option to option, basically picking out recommended editors from the forums and Google hits. I tried out ConTEXT. I rigged my Visual Studio with nFringe. For the longest time I used the license-free version of WOTgreal, which was brimming with features but had been made for UT2004. Then I switched to Unreal X-Editor: still in development, but hardly an improvement.

On top of that, college recently forced me to abandon my PC for an on-the-go MacBook Pro. I couldn’t simply stop coding, so I had to look for alternatives on the Mac. Not surprisingly, there were none.

So I decided to do what any frustrated computer science student would do: make my own IDE.

## Why Choose Onyx?

Whether you work on the PC or the Mac, the Onyx experience won’t change. That’s because Onyx was build around the idea of cross-platform support. When you download the IDE you’ll notice that it isn’t an installer or a Mac disk image. It’s just a Java distributable. You place the folder where you want it, maybe create a shortcut to the JAR that you can stick in the Start Menu or Applications stack, and that’s it.

Onyx is far from becoming a robust “industry standard” editor, but feedback from the gaming community is sure to change that quickly. And not only do I welcome feedback, if there are any questions or problems I am willing to help solve them.

Oh, and did I mention...Onyx is free?

# THE MENU BAR

## File Menu

**New** (Ctrl-N or ⌘-N)

Opens a new tab labeled “Untitled” and switches focus to that tab.

**Open…** (Ctrl-O or ⌘-O)

Brings up the File Chooser dialogue. From here you can select a file to open. This will create a new tab for that file and switch focus to it.

**Close** (Ctrl-W or ⌘-W)

Closes the currently selected tab. If changes have been made to the tab since its last save, Onyx will prompt the user to save changes (Yes), discard changes (No), or continue editing the tab (Cancel).

**Save** (Ctrl-S or ⌘-S)

Writes the currently selected tab to disk. If the tab has never been saved before (i.e. is “Untitled”), this will prompt the “Save As…” action instead.

**Save As…**

Brings up the File Saver dialogue. The saved file extension defaults to .uc, however note that Onyx can save any text file (.txt, .java, etc.) – just change the extension.

**Exit** (Ctrl-Q or ⌘-Q)

Quits Onyx. If unsaved changes have been made to any tabs, Onyx will prompt the user as in the “Close” action before quitting.

## Edit Menu

Onyx supports the following simple text editing commands:

**Undo** (Ctrl-Z or ⌘-Z)

**Redo** (Ctrl-Y or ⌘-Y)

**Cut** (Ctrl-X or ⌘-X)

**Copy** (Ctrl-C or ⌘-C)

**Paste** (Ctrl-V or ⌘-V)

**Select All** (Ctrl-A or ⌘-A)

**Find…** (Ctrl-F or ⌘-F)

Brings up the Find dialogue. For more information on what this can do, visit “Find and Replace” in the **Advanced Features** section.

**Find Next** (F3)

If there is something currently in the Find dialogue, this will find its next instance. Otherwise it will open the Find dialogue.

**Replace…** (Ctrl-R or ⌘-R)

Brings up the Replace dialogue. For more information on what this can do, visit “Find and Replace” in the **Advanced Features** section.

**Go To…** (Ctrl-G or ⌘-G)

Brings up the Goto Line dialogue. Enter a line number, and Onyx will jump to that line number for you.

**Tabs Submenu** (More info: “Tabs and Termination” in **Advanced Features**)

* **Set tab size**: Brings up an Input dialogue. Input the tab size (in spaces) you want to use.
* **Emulate tabs with spaces**: If this is selected, when you hit Tab Onyx will insert spaces instead of a tab character. The number of spaces inserted depends on your tab size.
* **Convert tabs to spaces**: Converts all tabs in the current document to spaces.
* **Convert spaces to tabs**: Converts all space blocks in the current document to tabs.

**Line Termination Submenu** (More info: “Tabs and Termination” in **Advanced Features**)

Select either PC style line termination (CR+LF, \r\n, or 0x0D0A) or UNIX style line termination (CR, \r, or 0x0D) to apply to the current document.

**Complete Code** (Ctrl-Space)

Calls the autocomplete system on the word or phrase you are currently typing. For more information on what this can do, visit “Autocomplete” in the Advanced Features section.

## View Menu

**Line Wrap**

Toggle line wrap (this selection is applied to all open documents).

**Line Numbers**

Toggle showing line numbers (this selection is applied to all open documents).

**White Space**

Toggle showing white space (this selection is applied to all open documents). If selected, you will see tabs as gray arrows and spaces as small midline gray dots.

**Highlight Current Line**

Toggle highlighting the current line (this selection is applied to all open documents). If selected, you will see a yellow bar over the line your cursor is currently on.

**Code Folding**

Toggle code folding (this selection is applied to all open documents). If selected, you will see prompts to fold bracketed sections of code to the right of the line numbers.

## Help Menu

**Contents…**

|  |
| --- |
| NOTE: This feature is currently a **Work In Progress** |

Launch the Onyx Help program.

**About Onyx UnrealScript Editor**

Brings up the About dialogue.

## Themes Submenu (View 🡪 Set Theme)

|  |  |  |
| --- | --- | --- |
| Onyx (Default) | | |
| Amazon | Apple Classic | Apple Modern |
| Apple Pink | Blue Lake | Desert |
| Emerald | Mint | Onyx Gold |
| Onyx Lemon | Onyx Sand | Onyx Sky |
| Sunshine | Tropical | Windows |

# THE WORK AREA

## Class Tree w/UnCodeX

I’ve been working with UDK ever since it came out in 2009. Although I’ve messed around with nearly every part of the engine, my specialty has always been programming: before learning UnrealScript I had taught myself everything from TI-BASIC to C/C++ to ActionScript.

**Basic UDK Integration**

Now that you have an account, the next step is to get it working with UDK. If you already have experience with UDK’s TCPLink system this should be a trivial exercise, you can get right to coding and have a workable system in a few hours.

But wait. Why do that when I’ve already done it for you? That’s right: download the handy little “UDK Starter Kit” from <http://willyg302.wordpress.com/gemini/> and you’ll have access to the full sources of a basic system using Gemini in UDK!

This system is far from complete, though (it’s called “Starter Kit” for a reason). What it does is fetch and parse the messages from your Gemini account given a password and field size and print them to the log. In order to actually do something useful, you WILL have to do a bit of coding. But that’s a topic for later on in this documentation.

**How Gemini Works**

Before diving into UnrealScript, step back for a moment. You’ll need to know how Gemini works.

UDK is an amazingly complex beast, but in the Internet department it’s rather primitive. It communicates strictly through POST/GET requests, which are the simplest ways that a computer can communicate with a server. A POST request sends data to a site, and a GET request gets data from a site. It’s that simple.

Therefore, Gemini has to operate with this in mind. When UDK gives a POST request to Gemini, Gemini takes the data from the request and adds it to your account. Options for receiving data are extremely limited: you can only specify a single string (called “content”) to send to Gemini.

When UDK sends a GET request, it actually sends it to a special page on Gemini called “service.jsp”. This page is a printout of the visible messages in your account, and it looks something like this:

<@>Test 2 (again) <@>Test of the better interface <@>JFrame Extended Trial 1 <@>Trial with JFrame <@>cuz i have the pass! :D <@>had to be me <@>aww snap who posted donuts <@>Message 7 <@>Message 6 <@>donuts

UDK receives the contents of this page in a single uninterrupted string, and it then parses the string to retrieve your messages.

You may be thinking, “Well geez, there’s not much you can do with string messages!” But you would be wrong. With a little bit of creativity, it’s not hard to see how the entire state of a complex FPS can be saved in the space of 10 or so strings. For example, you could just allocate the first string to how many people are online: “14172”. Or if you wanted to handle individual game modes, the string could be “14172,423,4332,567,3123” and that would mean “14172 total players, 423 playing CTF, 4332 playing Deathmatch...” and so on.

The system is rather simple, and that is a major benefit. Because UDK is not bogged down with backend SQLs and the like, it can instead focus its power on another thing: your game. All it does is send data to Gemini, and get some data back.

**IN-DEPTH GUIDE**

**The Field**

One of the most important concepts to understand is your account’s field: the messages (individual strings) that are visible on your “service.jsp” page.

Gemini saves messages in a string list called Messages, and it saves it in reverse order. When you add a new message to your account, it goes into Messages(0) and all the old messages get pushed back one; Messages(4) is now Messages(5), and so on. Regardless of whether the message is visible or not, it still exists in Messages until it is cleared.

However, there is another variable unique to your account called FieldSize. This determines the number of messages that are VISIBLE. Only visible messages get sent to UDK, so it is very important to make sure that your FieldSize is appropriate for your game (we’ll get to setting it later).

What Gemini does is set a variable X=FieldSize. Then it displays the first X messages from the list Messages onto “service.jsp” for UDK to grab. For example, with X=7 our previous example would be:

<@>Test 2 (again) <@>Test of the better interface <@>JFrame Extended Trial 1 <@>Trial with JFrame <@>cuz i have the pass! :D <@>had to be me <@>aww snap who posted donuts

And with X=3 it would be:

<@>Test 2 (again) <@>Test of the better interface <@>JFrame Extended Trial 1

**Using URLs**

UDK communicates with Gemini via URL. More specifically, UDK sends a request via this format:

http://geminionlinegs.appspot.com/service.jsp?content=your+message+here&pass=donuts&code=e34b56c

The question mark allows us to pass data in the form of parameters to the site. So what this does is send three variables, “content”, “pass”, and “code” to the “service.jsp” page. In this case, assuming your account password was “donuts” and the first 7 characters of your validation code were “e34b56c”, you would have just posted a new message (namely “your message here”) to the field.

The “pass” parameter is required for you to get the data you need, since it is the only way for Gemini to know what account to access. If you do not provide a “pass” parameter the “service.jsp” page will simply read “ERROR: Account not found.” (It would do the same if you provide a password for an account that doesn’t exist).

The “code” parameter is also required as a checker to make sure only the owner of the account can access it. If you do not provide this parameter, Gemini will throw an error. It will also do this if you provide the wrong code.

However, the “content” parameter is optional. If you do not provide it, Gemini simply displays the current messages and sends them to UDK – the equivalent of a GET request.

**Special Service Functions**

The service includes several functions that may be called from the “content” field, and which perform operations other than POST requests. Note that these functions are case-sensitive and exclusive to the service (that is, don’t post any message that includes “flush”, “replace”, etc., or else it will be interpreted as a function).

**flush#**

Deletes messages that are older than # messages old. To flush all messages, use “flush0” (warning: this would likely cripple the database on the UDK side). Your UDK should automatically handle flushing old messages at every session end, so this function may be redundant if used.

**-EX**: “flush5” would keep only messages 0-4 in the account, since 0,1,2,3,4 counts as 5 messages (with 0 as base)

**replace#,X**

Replaces message # in the service with a new message X. # must be an integer less than the field size of your account, where 0 is the newest (top-most) message when viewing the “service.jsp” page. This is extremely useful since it doesn’t push old messages down the stack.

**-EX**: “replace7,Hello World” would replace the 8th message in the field with “Hello World”

**delete#**

Deletes message # in the service (which pushes all messages below it up in the stack). Currently the message must be visible to be deleted, and so # must be an integer less than the field size, with 0 being the top-most message when viewing your account’s “service.jsp” page.

**-EX**: “delete8” deletes the 9th message in the field and pushes all messages below it up

**<@>**

Not really a function, but UDK’s way of parsing Gemini’s string dump into individual messages. Notice that each message you POST is appended to <@> which allows it to be split at that point. So, it is possible to hold more messages than the field size by putting <@> into your POST.

**-EX**: Posting “hello<@>world<@>!” would cause UDK to receive 3 messages, “hello”, “world”, and “!”

**setfieldsize#**

Sets your account’s field size to #. # must be an integer between 1 and 50; the maximum is 50 to avoid slowing down UDK unnecessarily when doing multiple GET requests. The default is 10, lightweight games can generally go 3-5, MMORPG-type games may need 30+.

**-EX**: “setfieldsize15” would set the field size of your account to 15 – “service.jsp” would display 15 messages

**perform#,op$,num&**

Performs a basic (+, -, \*, /) operation on message # using operation $, where ($=1) is add, ($=2) is subtract, ($=3) is multiply, and ($=4) is divide. The other input is an integer &. The message # must be an integer, and the resulting value is always an integer.

**-EX**: “perform2,op3,num5” would multiply the 3rd message in the field by 5 and store the result to the same message

**append#,X**

Appends string X to the end of message #. # must be an integer less than the field size of your account, where 0 is the newest (top-most) message when viewing the “service.jsp” page. This is a faster way to update messages when the entire string does not have to be deleted.

**-EX**: If message 4 is “Hello World”, then “append3,!” would replace it with “Hello World!”

**terminate**

Deletes your account from the system. If you decide to quit using Gemini, please send this message to your account to terminate it, or email me your password so that I can do it. Warning: once you terminate your account, you can never recover its data. Use with extreme caution.

**-EX**: Do you really need one?

If you think of a function that should be included in Gemini, email me an explanation of the function and its purpose. If I deem it useful enough I will add it!

# RANDOM OTHER STUFF

## Contact Me

You can contact me in one of the following ways:

* Send me an email at [willyg302@gmail.com](mailto:willyg302@gmail.com)
* Send me a YouTube message (my channel: [http://www.youtube.com/user/willyg302](http://www.youtube.com/user/willyg302/featured))
* Comment somewhere on my blog

Please use a descriptive subject line and assume that I’m incredibly stupid when explaining your problem. Allow several days for me to respond.

## Boring Legal Stuff

Copyright © 2012 WillyG Productions. All rights reserved.

Onyx IDE is made available under the Onyx License, which is packaged with the Onyx distribution. Please review this license before using Onyx. By using Onyx, you agree to the terms and conditions set forth in the Onyx License.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY, without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the Onyx License for more details.

## The End

Happy coding!