Neanderthal Pong Beta 2.0

Sun Sep 13 02:33:08 MDT 1992

Neanderthal Pong is a take-off on the traditional video game, Pong. Pong was the first real video game which was widely accepted and is therefore the inspiration to most videoheads like myself. Of course we have advanced to much higher levels of complexity with interactive computer graphics and artificial intelligence. But to start off, here is Neanderthal Pong, a primitive but essential game in the development of entertainment systems.

Playing Neanderthal Pong (NPong)

Release Beta 2.0 of Neanderthal Pong automatically starts a game. The reason is that NPong will eventually support demo mode, a simple yet fun-to-watch game where it just can't lose. It just vertically follows the ball around. So where the ball goes the paddle goes.

You can play this starting game, or better yet click the New Game button below Mr. Neanderthal (Actually the image is from Digital Webster and represents a Telamon, which is not at all a Neanderthal but a figure used on supporting columns. It looked good, so it was used.)

There are many different ball and paddle sizes as well as speeds (delta X) for you to use. Select them from the Neanderthal Pong menu. Warp speed with a Prenatal ball and a Ping Pong Paddle for a Monkey Spanker. You go figure. These are just names that came to mind while programming this thing. The bottom line is when the ball is moving fast, and the ball is small, and the paddle is small the game is going to be difficult. That is the point. You can make it as easy or as hard as you want.

The high score is kept and updated after each game. There is no current naming, so whoever gets the high score just has to be happy with their accomplishments rather than fame. This may change in future releases, then again it may not.

Source Code

This program was a 1.25 day hack which I would highly recommend you USE FOR REFERENCE AT YOUR OWN RISK. It is about the worst code I have written in years. I did this to show people quickly that animation on the NeXT using HyperCube is possible, and to basically use this as a gauge to speculate on using HyperCube as a front end to the TADS system (Text Adventure Developer System, by High Energy Software in California). My basic feeling is that HyperCube can and should be considered in such efforts (via the Externals Interface) to add sound and graphics support to already existing software. Yes HyperCube is slow, but then again 200 MIP machines are just around the corner. C'est La Vie.

Most of the script handlers/methods are located in the private layer to the page which the game screen resides. There are no shared layers. The scripts which are on the page layer are mostly 'public:' handlers, taking advantage of the fact a message is sent from an object to a page and then the message is sent through all the page layers (shared and private) and any public: handler which it matches will pick up the message and deal with it. It is a very subtle and important issue that I am going to focus the next release on, completely redesigning Neanderthal Pong to exploit and explain this point. Keep tuned for the next release.

Objects of Desire

No scaling of the images was performed for this release. It was tried, and the speed of such scaling proved to be detrimental to the over all effect of the game (which usually cause one to fall asleep or close the document within 1 minute of initial use).

The paddle images were ripped off from the NeXTstep environment. As you may have noticed they are just scrollview knobs. I swiped them from the Workspace Mangler. The ball images were generated using DisplayTalk from Adobe Systems. I highly recommend you purchase the DT system if you are into any type (and I mean ANY) of Display PostScript Programming. It is a wonderful and powerful aid in debugging DPS files. I used DT to generate the SPHERE file which is supplied with the DT system. I changed the diameter parameter and rendered the images. That simple!!!

I used Icon.app and Grab.app (NS 3.0 supplied software) to do the image grabbing and transparency filling. These two programs are essential if you are doing any type of image manipulation on the NeXT and do not have the inclination or funds to purchase the excellent programs like Image and PixelMagician. Image from Appsoft is about as good as an image manipulation program as you will get on the NeXT for a couple of years. I highly recommend you spend the money if you have it to spend...

Programming Explanations

The page which the game face is located, really does nothing. The layer is where all the stuff happens. This was by accident, not choice. I usually start programming common functionality scripts at the layer level, and then go to the page and then to shared layers if I have to write methods that are shared between many different pages. The layering in HyperCube is analogous to the card and background in HyperCard, but much more expansive. It also allows you to group objects naturally, since there are an unlimited number of layers allowed in a page.

The public: idle handler is where most of the action occurs during the event of a game. There are too many global variables used to keep track of what is going on where. Check it out. It sucks. But hey, it works.

New Paradigm

For any seasoned HyperHead programmer, it has been a long time coming to expand the language syntax of HyperTalk. This was somewhat dealt with in the externals of HyperCard (XCMDs and XFCNs) but really didn't deal with the need for an extension which supports message passing directly to objects, which gave the programmer the feeling they were dealing with true object, not just a hierarchy of action message paths. So, to help solve this paradigm dogma there were extensions made to the CubeTalk language.

They are not unlike the additions made to Obj-C from the C Language. The support for brackets and nesting has been added, and most of it has been implemented. The next couple of releases should prove to be more fully implemented. Where before now you might have the following

<u>Object "Wowzer" (a button)</u>

```
on doSomethingWierd
    doSomethingWackoAndStrange
end doSomethingWierd

on doSomethingElse
    doSomethingOtherThanWhatYouThink
end doSomethingElse
```

<u>Object "Dude" (a button)</u>

```
on mouseUp
    send "doSomethingWierd" to button "Wowzer"
    send "doSomethingElse" to button "Wowzer"
end mouseUp
```

You can now have this read

<u>Object "Wowzer" (a button)</u>

```
function doSomethingWierd
    doSomethingWackoAndStrange
    return ( me )
end doSomethingWierd

function doSomethingElse
    doSomethingOtherThanWhatYouThink
    return ( me )
end doSomethingElse
```

<u>Object "Dude" (a button)</u>

```
on mouseUp
    [[ button "Wowzer" doSomethingWierd ] doSomethingElse ]
```

end mouseUp

This may seem like a trivial example, and in the big picture it may be. But the implications are wide spread. This added syntax (similar to Obj-C, SmallTalk, Self and other object languages) gives you the programmer, and the user a stronger object paradigm to work with over that of traditional programming languages and even HyperTalk. The syntax is more terse and easier to read. Thus it is easier to understand and maintain. Also this means it is easier to modify because it allows the programmer to encapsulate the data and functionality of an object. The bottom line is better, cleaner, and easier to modify/maintain programs.

I thank Thoughtful Software for implementing my idea of using message passing in this context. It turned out to be a 4-6 hour job to get the parser and compiler to handle this change in syntax. This is a phenomenal accomplishment and helps support claims that NeXTstep and Obj-C are about the best things since sliced bread (oh, can't forget the AppKit. It is the low salt butter).

[Thoughtful Software's note: The new syntax described above is only partially implemented in the Beta02 release. You can send messages to an object using [<object> <message>] *but you can't return an object from a function, so the type of message nesting illustrated above can't be done yet. We expect to add that capability in the Beta03 release.]*

Summary

This example is a mess right now. But will be cleaned up by the next release. I was pushed and such I got it out the door in semi-acceptable shape. Take this all with a large grain of salt (ie. a salt lick if you can) and use what you want. Neanderthal Pong should prove to be fun for the young and old at heart. I enjoyed doing it, and learned a lot about the following developments

• idle messaging and altering the idleDelay values
• public: (soon to be called page:) handlers in layers
• layers and how messages are passed through them to offer HyperCard compatibility
• menus: creation, deletion, resetting, menuMessaging and CubeCrashing ;')
• more about encapsulation of objects and good object design (not seen here at this time)
• Life the universe and everything

About the Author

If there are any questions about programming HyperCube, CubeTalk, HyperCard or any other system (ie, Plus, ToolBook, SuperCard) and you have email, direct your questions and answers to **<hypercube-prog@liveware.com>**. This is a mailing list alias which is maintained by moi, and will help facilitate the development of your documents, and hopefully expedite your learning process. To be added to this list, send a mail message to **<hypercube-prog-request@liveware.com>** asking to be added to the list or deleted from the list (should you ever want to leave us). PLEEEEAZE, do NOT send requests of 'addition to' and 'removal from' the list to the list itself. Others are not going to want to read it.

I am currently working on a complete guide about using and programming HyperCube. It is due out early next year and will cover everything from beginning users needs to the most obscure problems found in HyperCube programming. Please email me if you have documents or chapters you would like to work on for the book. I do have some areas where I won't have the time to complete them by the deadline. Contact me at <todd@liveware.com> for more information.