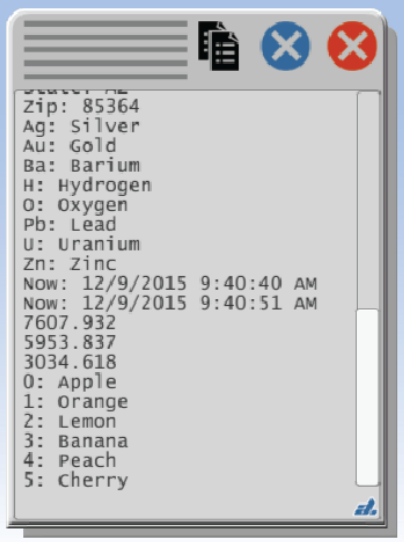
dynamic arts

DABug

Debugging Message Window for Unity

Must-have dev tool!
2D/3D/UI/Combo scenes
Active between scenes



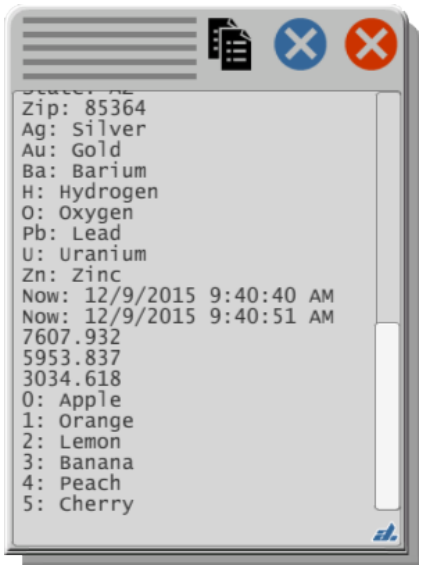
Dynamic Arts DABug v4.2
Debugging Message Window for Unity
Documentation

http://DynamicArts.com

dynamic arts

DABug Overview

The DABug Debugging Message Window
Is a great visual development tool which displays your
Debugging messages in a draggable window inside your scene
Instead of the Debug.Log console window.



It holds up to 2000 lines of rolling text, which can be copied to the Clipboard and pasted anywhere.

It's quick and easy!!!

Just drop the prefab anywhere on your project.

Instead of your standard debug message:
```
Debug.Log(message);
```
We use:
```
DABug.Log(message);
```
OR QUICK-TYPE COMMAND:
```
a.s(message);
```

Also includes simple code keywords for:
Copying text
Opening/Closing the window
Positioning the window
Clearing the window text

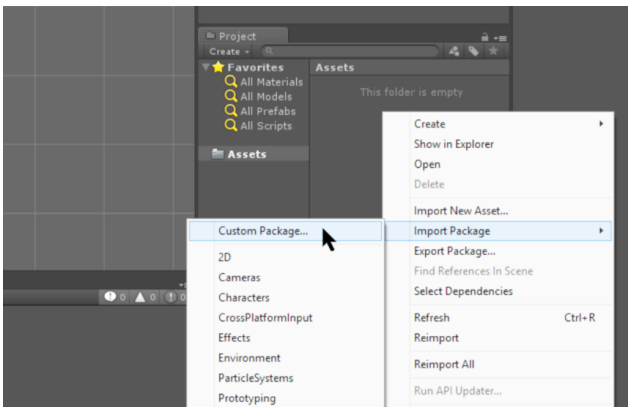
See **Appendix B: Code Samples** at the end of this document.

Note
This Unity package was built with Unity 5.5. There are a few new programming changes that will not work in previous versions.

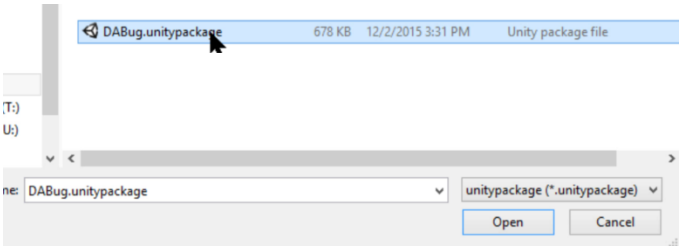
DABug Installation 1-2-3

If you are installing from the Unity Asset Store, continue to Step 3.

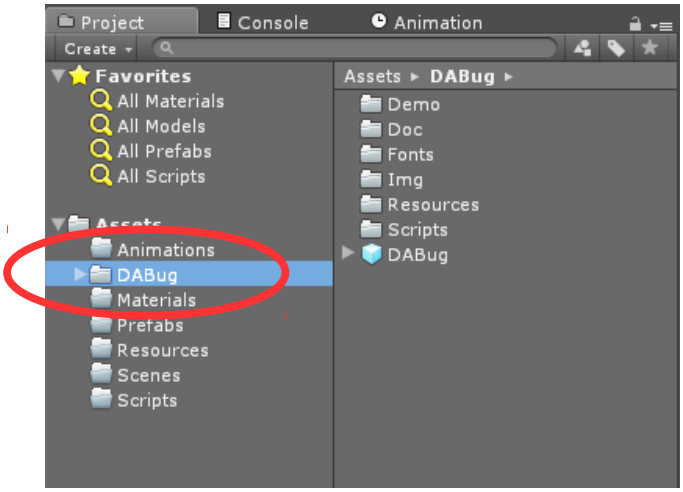
1. Manual installation: Download the DABug Custom Package



2. From the Assets folder, right-click → Import Package → Custom Package

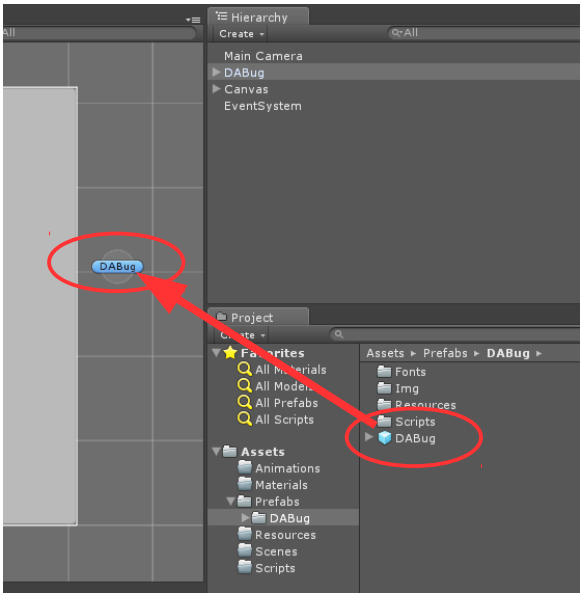


3. Choose the DABug Unity Package from your download directory. Import everything listed in the package.

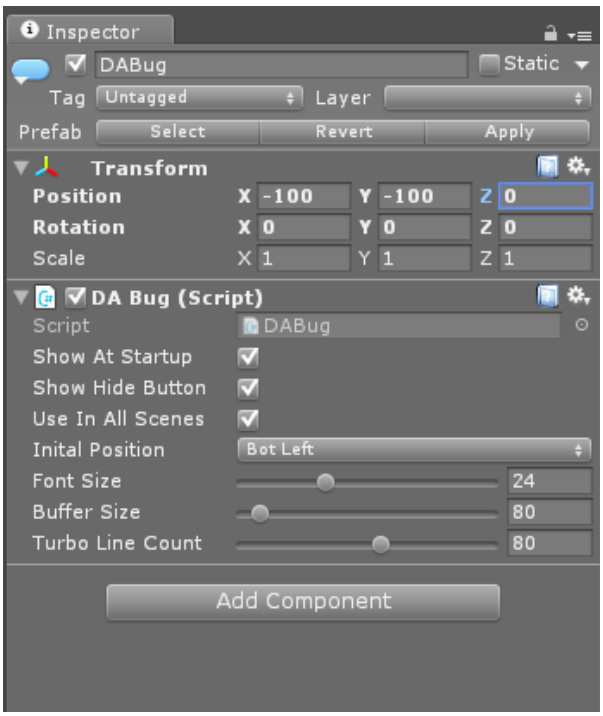


One folder will be created:
DABug: Holds the main DABug window prefab. You may want to move this folder to your **Prefabs** folder.

Inside, the **Demo** folder holds the demonstration scenes.
You can delete the **Demo** folder if you don't need the demo scenes.

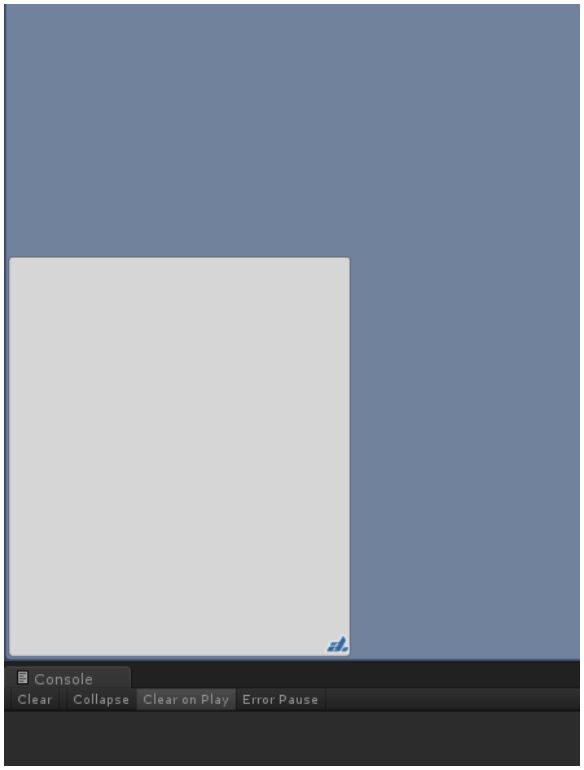


From the DABug folder, simply drag the DABug prefab **anywhere into your scene**. The object will appear as a small icon.

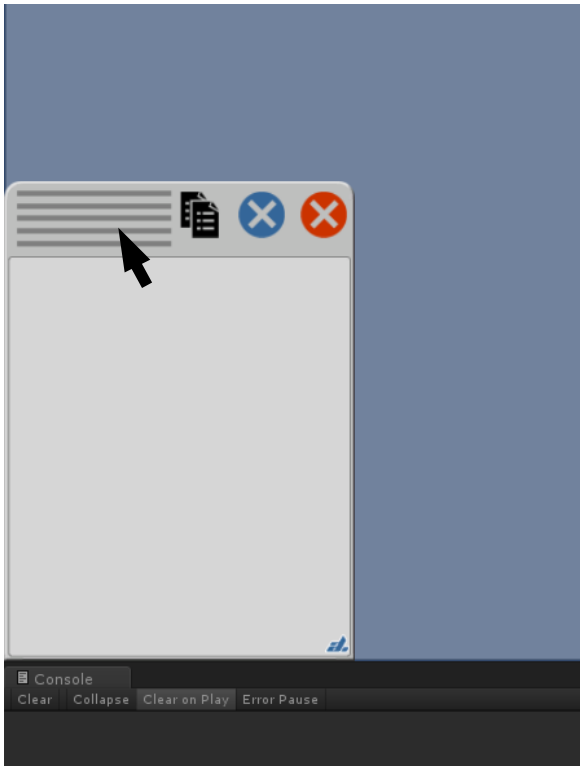


You may adjust the DABug object properties in the Inspector, if desired.
See **Appendix A** for property descriptions.

DABug Usage continued



When the scene is played,
the DABug window will appear in the corner.

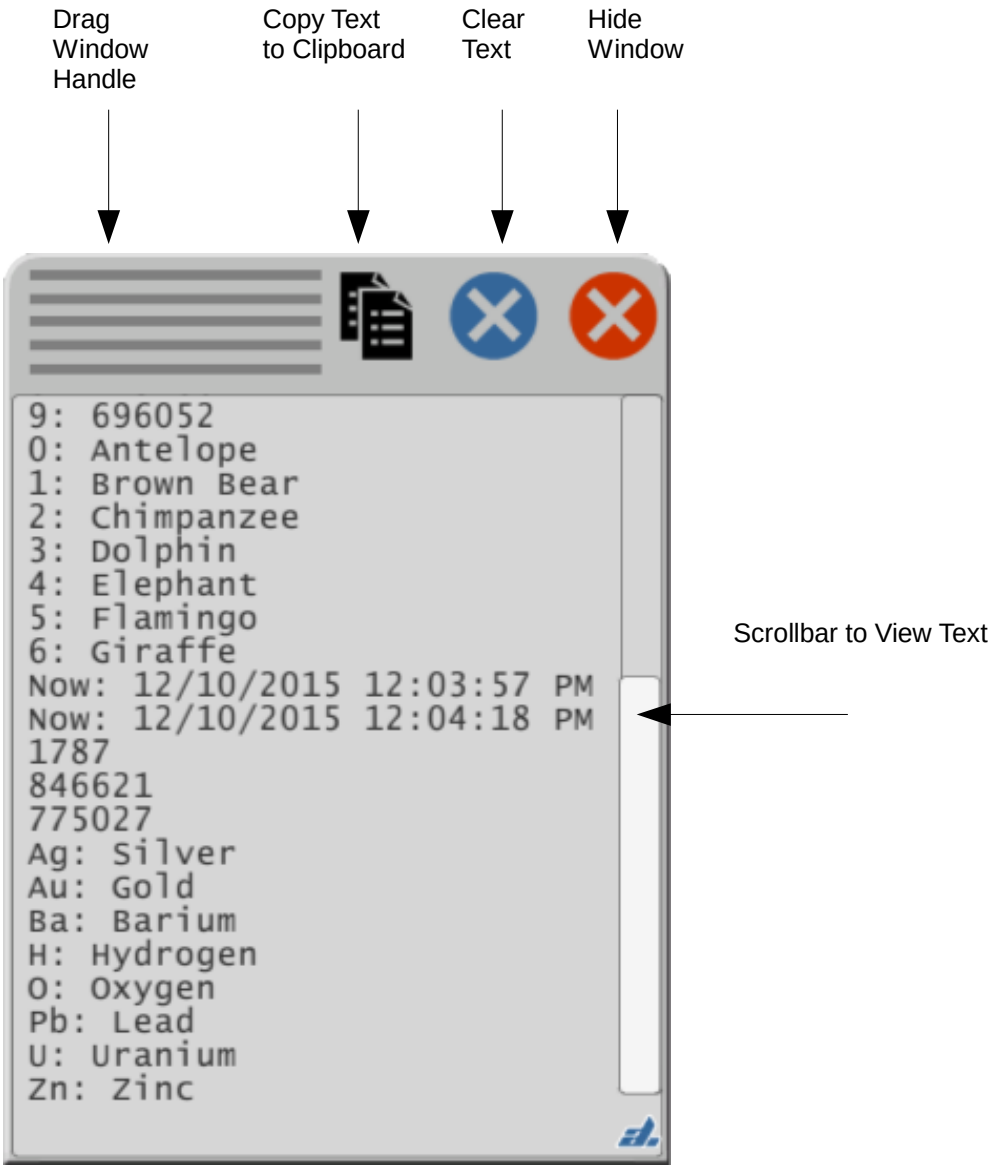


When the mouse rolls over the DABug window,
the top expands, revealing controls to:
Drag, Copy, Clear and Remove the window.

Your debugging messages can appear in the window by using the **DABug.Log()** or **a.s()** command.
See **Appendix B: Code Samples** for examples.

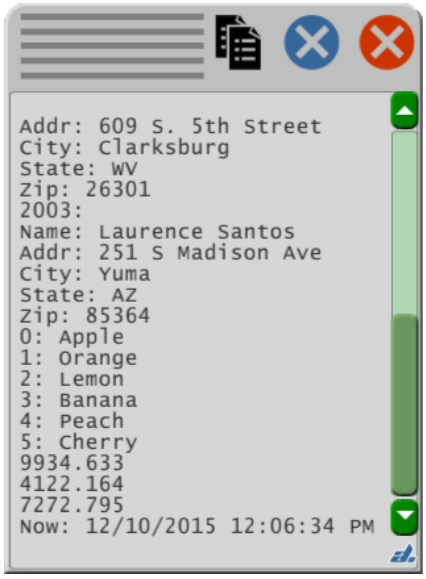
If a message line exceeds 100 characters in length, it is split into multiple lines.

Mouse/Finger Controls During Execution



Turbo Mode

Normal Mode



Turbo Mode: Scrollbar turns green.

Normal Mode: The window text will scroll normally, line by line.
Turbo Mode: The window text will scroll page by page.

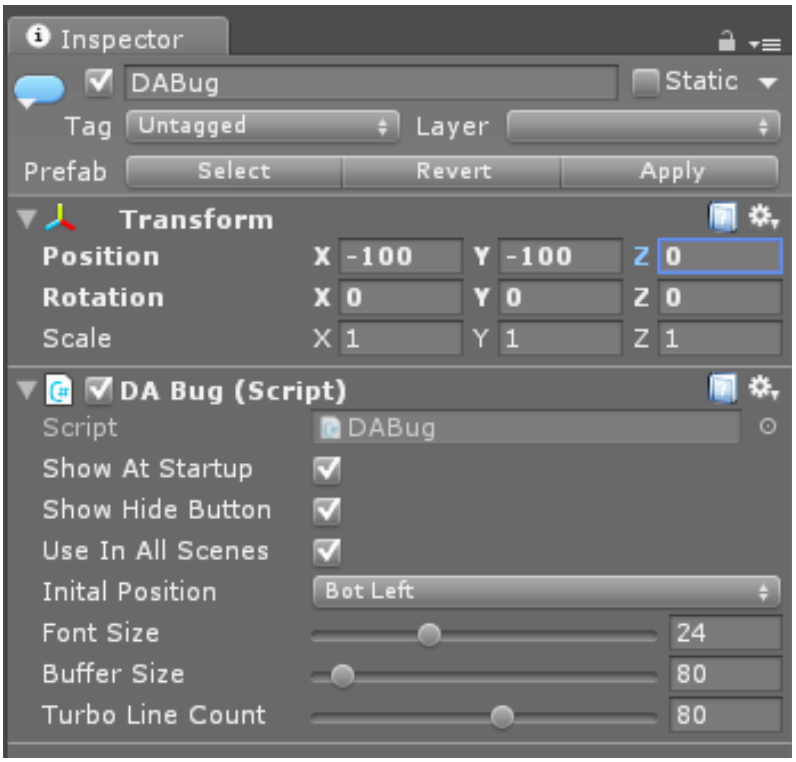
The DABug window will enter Turbo Mode when the Turbo Line Count property is exceeded.
See **Appendix A: DABug Prefab Inspector Properties** for more information.
Turbo Mode changes the text scrolling behavior from line-by-line to page-by-page.

Text paging is employed to reduce the number of vertices needed to draw the object.
This keeps the object extremely lightweight and will not slow down the render time in your project.

Note!

Your other objects/code in your projects may also increase the vertex count.
If you are finding that setting a DABug message causes a vertex error (too many vertexes),
Simply lower the **Turbo Line Count parameter in the inspector**. This will keep the vertex count low.

Appendix A: DABug Prefab Inspector Properties



Show At Startup
The DABug window is active upon project execution.
To show or hide the window, use the DABug.Log("#show#") or DABug.Log("#hide#") commands.

Show Hide Button
Display the "Hide" button.
Hiding this prevents accidental disabling of the DABug window.

Use in All Scenes
Allow the DABug window to persist in all scenes.

Initial Position
Upon execution, display the DABig window in the selected corner.
The window is always able to be dragged with the mouse or finger.

Font Size
Window text font size
Min: 18, Max: 36, Default: 24

Buffer Size
Total number if text lines hold
Min: 10, Max: 2000, Default: 80

Turbo Line Count
# of scrollable lines before window enters Turbo mode.
Min: 30. Max: 120, Default: 80

```
// IMPORTANT !!!
// At the top of your script, include the DArts namespace
using DArts;


// If a message line exceeds 100 characters in length, it is split into multiple lines.


// Usage
DABug.Log(my_str);

// OR use the quick-type usage:
a.s(my_str);



// String
string my_str = "Have a nice day!";
DABug.Log(my_str);
or
a.s(my_str);

Have a nice day!



// Integer
int my_int = 123456;
DABug.Log(my_int);

123456



// Float
float my_flt = 654.321f;
DABug.Log(my_flt);

654.321
```

Collections

While the DABug() command quickly displays simple strings and numbers, it can also display a limited amount of other simple variable types, without having to iterate through the collections with code.

```
// String Array
string[] my_str_arr = {"Antelope", "Brown Bear", "Chimpanzee", "Dolphin", "Elephant",
"Flamingo", "Giraffe"};
DABug.Log(my_str_arr);
```

```
0: Antelope
1: Brown Bear
2: Chimpanzee
3: Dolphin
4: Elephant
5: Flamingo
6: Giraffe
```

```
// You can always loop through the collection yourself with code
foreach(string item in my_str_arr) {
     DABug.Log(item);
}
```

```
Antelope
Brown Bear
Chimpanzee
Dolphin
Elephant
Flamingo
Giraffe
```

```
// Integer Array
int[] my_int_arr = new int[10];
for (int i=0; i<=9; i++) my_int_arr[i] = Random.Range(0,1000000);
DABug.Log(my_int_arr);
```

```
0: 792042
1: 179577
2: 141550
3: 536760
4: 757294
5: 282339
6: 556744
7: 373804
8: 536689
9: 738625
```

```
// List<string>
List<string> my_list_str = new List<string>();
my_list_str.Add("Apple");
my_list_str.Add("Orange");
my_list_str.Add("Lemon");
my_list_str.Add("Banana");
my_list_str.Add("Peach");
my_list_str.Add("Cherry");
DABug.Log(my_list_str);
```

```
0: Apple
1: Orange
2: Lemon
3: Banana
4: Peach
5: Cherry
```

```
// Dictionary<string, string>
Dictionary<string, string> my_dct_ss = new Dictionary<string, string>();
my_dct_ss.Add("Ag", "Silver");
my_dct_ss.Add("Au", "Gold");
my_dct_ss.Add("Ba", "Barium");
my_dct_ss.Add("H", "Hydrogen");
my_dct_ss.Add("O", "Oxygen");
my_dct_ss.Add("Pb", "Lead");
my_dct_ss.Add("U", "Uranium");
my_dct_ss.Add("Zn", "Zinc");
DABug.Log(my_dct_ss);
```

```
Ag: Silver
Au: Gold
Ba: Barium
H: Hydrogen
O: Oxygen
Pb: Lead
U: Uranium
Zn: Zinc
```

```
// Dictionary<string, object>
Dictionary<string, object> my_dct_so = new Dictionary<string, object>();

Dictionary<string, string> my_record_0 = new Dictionary<string, string>();
my_record_0.Add("Name","Dexter Campbell");
my_record_0.Add("Addr","110 S. Wabash");
my_record_0.Add("City","Chicago");
my_record_0.Add("State","IL");
my_record_0.Add("Zip","60603");
Dictionary<string, string> my_record_1 = new Dictionary<string, string>();
my_record_1.Add("Name","Rosa Lucas");
my_record_1.Add("Addr","166 Central Ave.");
my_record_1.Add("City","Charleston");
my_record_1.Add("State","SC");
my_record_1.Add("Zip","29406");
Dictionary<string, string> my_record_2 = new Dictionary<string, string>();
my_record_2.Add("Name","Molly Weber");
my_record_2.Add("Addr","609 S. 5th Street");
my_record_2.Add("City","Clarksburg");
my_record_2.Add("State","WV");
my_record_2.Add("Zip","26301");
Dictionary<string, string> my_record_3 = new Dictionary<string, string>();
my_record_3.Add("Name","Laurence Santos");
my_record_3.Add("Addr","251 S Madison Ave");
my_record_3.Add("City","Yuma");
my_record_3.Add("State","AZ");
my_record_3.Add("Zip","85364");

my_dct_so.Add("2000", my_record_0);
my_dct_so.Add("2001", my_record_1);
my_dct_so.Add("2002", my_record_2);
my_dct_so.Add("2003", my_record_3);
DABug.Log(my_dct_so);

2000:
Name: Dexter Campbell
Addr: 110 S. Wabash
City: Chicago
State: IL
Zip: 60603
2001:
Name: Rosa Lucas
Addr: 166 Central Ave.
City: Charleston
State: SC
Zip: 29406
2002:
Name: Molly Weber
Addr: 609 S. 5th Street
City: Clarksburg
State: WV
Zip: 26301
2003:
Name: Laurence Santos
Addr: 251 S Madison Ave
City: Yuma
State: AZ
Zip: 85364
```

```
// List<object>
List<object> my_list_obj = new List<object>();
my_list_obj.Add("Have a nice day!");
my_list_obj.Add(12345);
my_list_obj.Add(my_dct_ss);
my_list_obj.Add(my_str_arr);
DABug.Log(my_list_obj);
```

```
0:
Have a nice day!
1:
12345
2:
Ag: Silver
Au: Gold
Ba: Barium
H: Hydrogen
O: Oxygen
Pb: Lead
U: Uranium
Zn: Zinc
3:
0: Antelope
1: Brown Bear
2: Chimpanzee
3: Dolphin
4: Elephant
5: Flamingo
6: Giraffe
```

```
// Control commands

DABug.Log("#show#");
// or
a.s("#show#");
```

The DABug window is visible (active)

```
DABug.Log("#hide#");
```

The DABug window is hidden (inactive)

```
DABug.Log("#flip#");
```

The DABug window visibility is reversed (active/inactive)

```
DABug.Log("#clear#");
```

The DABug window text is cleared

```
DABug.Log("#copy#");
```

The DABug window text is copied to clipboard

```
DABug.Log("#bl#");
```

The DABug window is positioned to the screen bottom left

```
DABug.Log("#br#");
```

The DABug window is positioned to the screen bottom right

```
DABug.Log("#tl#");
```

The DABug window is positioned to the screen top left

```
DABug.Log("#tr#");
```

The DABug window is positioned to the screen top right