
/ This is \
\
\ ducksay! /

\
 \
 --
 >(')
)/
 /(
 / '----/
 \ ~== /
~~~~~

-----  
( But which Version? )  
-----

\  
 \  
 >()\_  
 (\_\_)\_\_

-----  
( v2.3 )  
-----

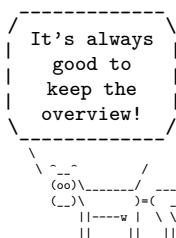
^--^  
 /--(oo) /  
 / \ (\_\_\_\_) /  
 | w----||  
 || ||

-----  
( by Jonathan P. Spratte )  
-----

/  
 .-----,  
 .'\_/\_|\_\`  
 /<::[0]8::>>\  
 |-----|  
 | | -==--== | |  
 | | ==--== | |  
 \ ||( )|::: | /  
 | ||( )|... | |  
 | |-----| |  
 | | \\_\_\_\_\_/ | |  
 / \ / \ / \ \  
 (\_\_\_\_) (\_\_\_\_) (\_\_\_\_)

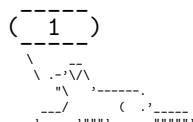
-----  
( Today is 2019-01-13 )  
-----

\ .\|//|\\|  
 \ |/\|/|/|/|/  
 /. '|\|/|/|/|  
 o\_\_ \_|//|/|\\|'



# Contents

|          |                                           |           |
|----------|-------------------------------------------|-----------|
| <b>1</b> | <b>Documentation</b>                      | <b>2</b>  |
| 1.1      | Downward Compatibility Issues             | 2         |
| 1.2      | Shared between versions                   | 2         |
| 1.2.1    | Macros                                    | 2         |
| 1.2.2    | Options                                   | 3         |
| 1.2.2.1  | Options for <code>\AddAnimal</code>       | 4         |
| 1.3      | Version 1                                 | 5         |
| 1.3.1    | Introduction                              | 5         |
| 1.3.2    | Macros                                    | 5         |
| 1.3.3    | Options                                   | 5         |
| 1.3.4    | Defects                                   | 6         |
| 1.4      | Version 2                                 | 7         |
| 1.4.1    | Introduction                              | 7         |
| 1.4.2    | Macros                                    | 7         |
| 1.4.3    | Options                                   | 7         |
| 1.5      | Dependencies                              | 12        |
| 1.6      | Available Animals                         | 12        |
| 1.7      | Miscellaneous                             | 14        |
| <b>2</b> | <b>Implementation</b>                     | <b>15</b> |
| 2.1      | Shared between versions                   | 15        |
| 2.1.1    | Variables                                 | 15        |
| 2.1.1.1  | Integers                                  | 15        |
| 2.1.1.2  | Sequences                                 | 15        |
| 2.1.1.3  | Token lists                               | 15        |
| 2.1.1.4  | Boolean                                   | 15        |
| 2.1.1.5  | Boxes                                     | 15        |
| 2.1.2    | Regular Expressions                       | 15        |
| 2.1.3    | Messages                                  | 15        |
| 2.1.4    | Key-value setup                           | 15        |
| 2.1.4.1  | Keys for <code>\AddAnimal</code>          | 16        |
| 2.1.5    | Functions                                 | 17        |
| 2.1.5.1  | Generating Variants of External Functions | 17        |
| 2.1.5.2  | Internal                                  | 17        |
| 2.1.5.3  | Document level                            | 19        |
| 2.1.6    | Load the Correct Version and the Animals  | 20        |
| 2.2      | Version 1                                 | 21        |
| 2.2.1    | Functions                                 | 21        |
| 2.2.1.1  | Internal                                  | 21        |
| 2.2.1.2  | Document level                            | 23        |
| 2.3      | Version 2                                 | 24        |
| 2.3.1    | Messages                                  | 24        |
| 2.3.2    | Variables                                 | 24        |
| 2.3.2.1  | Token Lists                               | 24        |
| 2.3.2.2  | Boxes                                     | 24        |
| 2.3.2.3  | Bools                                     | 24        |
| 2.3.2.4  | Coffins                                   | 24        |
| 2.3.2.5  | Dimensions                                | 24        |





```
\AddAnimal \AddAnimal[*][<options>]{<animal>}{ascii-art}
```

adds `<animal>` to the known animals. `<ascii-art>` is multi-line verbatim and therefore should be delimited either by matching braces or by anything that works for `\verb`. If the star is given `<animal>` is the new default. One space is added to the begin of `<animal>` (compensating the opening symbol). The symbols signaling the speech bubble's tail (in the `hedgehog` example below the two `s`) can be set using the `tail-symbol` option and only the first `tail-count` occurrences will be substituted (see [paragraph 1.2.2.1](#) for more about these options). For example, `hedgehog` is added with:

```
\AddAnimal[tail-symbol=s]{hedgehog}
```

```
{  s  .\|//||\||.
    s  |/\|/|||//|/|
      /. '|\|\|/||/||
    o  , |//|/||\||}'
```

It is not checked whether the animal already exists, you could therefore redefine existing animals with this macro.

`\AddColoredAnimal`    `\AddColoredAnimal<*>[<options>]{<animal>}<ascii-art>`

It does the same as `\AddAnimal` but allows three different colouring syntaxes. You can use `\textcolor` in the `<ascii-art>` with the syntax `\textcolor{<color>}{<text>}`. Note that you can't use braces in the arguments of `\textcolor`.

You can also use a delimited `\color` of the form `\bgroup\color{<color>}<text>\egroup`, a space after that `\egroup` will be considered a space in the output, you don't have to leave a space after the `\egroup` (so `\bgroup\color{red}RedText\egroupOtherText` is valid syntax). You can't nest delimited `\colors`.

Also you can use an undelimited `\color`. It affects anything until the end of the current line (or, if used inside of the `\text` of a delimited `\color`, anything until the end of that delimited `\color`'s `\text`). The syntax would be `\color{<color>}`.

The package doesn't load anything providing those colouring commands for you and it doesn't provide any coloured animals. The parsing is done using regular expressions provided by L<sup>A</sup>T<sub>E</sub>X3. It is therefore slower than the normal `\AddAnimal`.

`\AnimalOptions`    `\AnimalOptions{*}{\langle animal \rangle}{\langle options \rangle}`

With this macro you can set `\animal` specific `\options`. If the star is given any currently set options for this `\animal` are dropped and only the ones specified in `\options` will be applied, else `\options` will be added to the set options for this `\animal`. The set `\options` can set the `tail-1` and `tail-2` options and therefore overwrite the effects of `\duckthink`, as `\duckthink` really is just `\ducksay` with the `think` option.

### 1.2.2 Options

The following options are available independent on the used code variant (the value of the `version` key). They might be used as package options – unless otherwise specified – or used in the macros `\DucksayOptions`, `\ducksay` and `\duckthink` – again unless otherwise specified. Some options might be accessible in both code variants but do slightly different things. If that’s the case they will be explained in [subsubsection 1.3.3](#) and [subsubsection 1.4.3](#) for `version 1` and `2`, respectively.

$$\text{version}=\langle \textit{number} \rangle$$

With this you can choose the code variant to be used. Currently 1 and 2 are available.

( 3 )

This can be set only during package load time. For a dedicated description of each version look into [subsection 1.3](#) and [subsection 1.4](#). The package author would choose `version=2`, the other version is mostly for legacy reasons. The default is 2.

`<animal>` One of the animals listed in [subsection 1.6](#) or any of the ones added with `\AddAnimal`. Not useable as package option. Also don't use it in `\DucksayOptions`, it'll break the default animal selection.

`animal=<animal>` Locally sets the default animal. Note that `\ducksay` and `\duckthink` do digest their options inside of a group, so it just results in a longer alternative to the use of `<animal>` if used in their options.

`ligatures=<token list>` each token you don't want to form ligatures during `\AddAnimal` should be contained in this list. All of them get enclosed by grouping `{` and `}` so that they can't form ligatures. Giving no argument (or an empty one) might enhance compilation speed by disabling this replacement. The formation of ligatures was only observed in combination with `\usepackage[T1]{fontenc}` by the author of this package. Therefore giving the option `ligatures` without an argument might enhance the compilation speed for you without any drawbacks. Initially this is set to `'<>,'-`.  
**Note:** In earlier releases this option's expected argument was a regular expression. This means that this option is not fully downward compatible with older versions. The speed gain however seems worth it (and I hope the affected documents are few).

`no-tail` Sets `tail-1` and `tail-2` to be a space.

`say` Sets `tail-1` and `tail-2` as backslashes.

`tail-1=<token list>` Sets the first tail symbol in the output to be `<token list>`. If set outside of `\ducksay` and `\duckthink` it will be overwritten inside of `\duckthink` to be 0.

`tail-2=<token list>` Sets every other tail symbol except the first one in the output to be `<token list>`. If set outside of `\ducksay` and `\duckthink` it will be overwritten inside of `\duckthink` to be o.

`think` Sets `tail-1=0` and `tail-2=o`.

### 1.2.2.1 Options for `\AddAnimal`

The options described here are only available in `\AddAnimal` and `\AddColoredAnimal`.

`tail-count=<int>` sets the number of tail symbols to be replaced in `\AddAnimal` and `\AddColoredAnimal`. Initial value is 2. If the value is negative every occurrence of `tail-symbol` will be replaced.

`tail-symbol=<str>` the symbol used in `\AddAnimal` and `\AddColoredAnimal` to mark the bubble's tail. The argument gets `\detokenized`. Initially a single backslash.

```
( 4 )
 \ .-'\ \
  " \   \
  /    \ ( .)
 /-----\

```

### 1.3.1 Introduction

This version is included for legacy support (old documents should behave the same without any change to them – except the usage of `version=1` as an option, for a more or less complete list of downward compatibility related problems see [subsection 1.1](#)). For the bleeding edge version of `ducksay` skip this subsection and read [subsection 1.4](#).

The following is the description of macros which differ in behaviour from those of version 2.

options might include any of the options described in [subsection 1.2.2](#) and [subsection 1.3.3](#) if not otherwise specified. Prints an `<animal>` saying `<message>`. `<message>` is not read in verbatim. Multi-line `<message>`s are possible using `\\`. `\\` should not be contained in a macro definition but at toplevel. Else use the option `ht`.

options might include any of the options described in [subsection 1.2.2](#) and [subsection 1.3.3](#) if not otherwise specified. Prints an `<animal>` thinking `<message>`. `<message>` is not read in verbatim. Multi-line `<message>`s are possible using `\\`. `\\` should not be contained in a macro definition but at toplevel. Else use the option `ht`.

The following options are available to `\ducksay`, `\duckthink`, and `\DucksayOptions` and if not otherwise specified also as package options:

use `<code>` in a group right before the bubble (for font switches). Might be used as a package option but not all control sequences work out of the box there.

`body=<code>` use `<code>` in a group right before the body (meaning the `<animal>`). Might be used as a package option but not all control sequences work out of the box there. E.g. to right-align the `<animal>` to the bubble, use `body=\hfill`.

`align=valign`  
 use `⟨valign⟩` as the vertical alignment specifier given to the `tabular` which is around  
 the contents of `\ducksay` and `\duckthink`.

`msg-align=halign`  
 use *halign* for alignment of the rows of multi-line *message*s. It should match a **tabular** column specifier. Default is **l**. It only affects the contents of the speech bubble not the bubble.

`rel-align=<column>`  
 use *<column>* for alignment of the bubble and the body. It should match a `tabular` column specifier. Default is 1.

**wd**= $\langle count \rangle$  in order to detect the width the  $\langle message \rangle$  is expanded. This might not work out for some commands (e.g. `\url` from `hyperref`). If you specify the width using **wd** the  $\langle message \rangle$  is not expanded and therefore the command *might* work out.  $\langle count \rangle$  should be the character count.

**ht**= $\langle count \rangle$  you might explicitly set the height (the row count) of the  $\langle message \rangle$ . This only has an effect if you also specify **wd**.

```

-----
\ Ohh, no!
/-----\
 \ (.)_(.)
  (-----)
 /-----\
--\ ( ( ( ( ) ) /--
   \ \ \ \ / \ / \
    )_ / / \   / \ \ (

```

### 1.3.4 Defects

- no automatic line wrapping

```

-----
( 6 )
-----
 \ .-' \ \
  " \   '-----
   /   ( '-----
-----)-----)-----)-----)-----)

```

## 1.4 Version 2

### 1.4.1 Introduction

Version 2 is the current version of `ducksay`. It features automatic line wrapping (if you specify a fixed width) and in general more options (with some nasty argument parsing).

If you're already used to version 1 you should note one important thing: You should only specify the `version` and the `ligatures` during package load time as arguments to `\usepackage`. The other keys might not work or do unintended things and only don't throw errors or warnings because of the legacy support of version 1. After the package is loaded, keys only used for version 1 will throw an error.

### 1.4.2 Macros

The following is the description of macros which differ in behaviour from those of version 1.

---

|                       |                                             |
|-----------------------|---------------------------------------------|
| <code>\ducksay</code> | <code>\ducksay[⟨options⟩]{⟨message⟩}</code> |
|-----------------------|---------------------------------------------|

---

options might include any of the options described in [subsection 1.2.2](#) and [subsection 1.4.3](#) if not otherwise specified. Prints an `⟨animal⟩` saying `⟨message⟩`.

The `⟨message⟩` can be read in in four different ways. For an explanation of the `⟨message⟩` reading see the description of the `arg` key in [subsection 1.4.3](#).

The height and width of the message is determined by measuring its dimensions and the bubble will be set accordingly. The box surrounding the message will be placed both horizontally and vertically centred inside of the bubble. The output utilizes L<sup>A</sup>T<sub>E</sub>X3's coffin mechanism described in [interface3.pdf](#) and the documentation of `xcoffins`.

---

|                         |                                               |
|-------------------------|-----------------------------------------------|
| <code>\duckthink</code> | <code>\duckthink[⟨options⟩]{⟨message⟩}</code> |
|-------------------------|-----------------------------------------------|

---

The only difference to `\ducksay` is that in `\duckthink` the `⟨animal⟩`s think the `⟨message⟩` and don't say it.

### 1.4.3 Options

In version 2 the following options are available. Keep in mind that you shouldn't use them during package load time but in the arguments of `\ducksay`, `\duckthink` or `\DucksayOptions`.

`arg=⟨choice⟩`

specifies how the `⟨message⟩` argument of `\ducksay` and `\duckthink` should be read in. Available options are `box`, `tab` and `tab*`:

**box** the argument is read in either as a `\hbox` or a `\vbox` (the latter if a fixed width is specified with either `wd` or `wd*`). Note that in this mode any arguments relying on category code changes like e.g. `\verb` will work (provided that you don't use `\ducksay` or `\duckthink` inside of an argument of another macro of course).

**tab** the argument is read in as the contents of a `tabular`. Note that in this mode any arguments relying on category code changes like e.g. `\verb` will *not* work. This mode comes closest to the behaviour of version 1 of `ducksay`.

```

( 7 )
 \ .-'\ \
  " \   \
  /    \ ( .)
 /-----\

```



**tab\***

the argument is read in as the contents of a **tabular**. However it is read in verbatim and uses `\scantokens` to rescan the argument. Note that in this mode any arguments relying on category code changes like e.g. `\verb` will work. You can't use `\ducksay` or `\duckthink` as an argument to another macro in this mode however.

**b** shortcut for `out-v=b`.

`body=<font>` add `<font>` to the font definitions in use to typeset the `<animal>`'s body.

`body*=<font>`  
clear any definitions previously made (including the package default) and set the font definitions in use to typeset the `<animal>`'s body to `<font>`. The package default is `\verbatim@font`. In addition `\frenchspacing` will always be used prior to the defined `<font>`.

`body-align=<choice>`  
sets the relative alignment of the `<animal>` to the `<message>`. Possible choices are `l`, `c` and `r`. For `l` the `<animal>` is flushed to the left of the `<message>`, for `c` it is centred and for `r` it is flushed right. More fine grained control over the alignment can be obtained with the keys `msg-to-body`, `body-to-msg`, `body-x` and `body-y`. Package default is `l`.

`body-mirrored=<bool>`  
if set true the `<animal>` will be mirrored along its vertical centre axis. Package default is `false`. If you set it `true` you'll most likely need to manually adjust the alignment of the body with one or more of the keys `body-align`, `body-to-msg`, `msg-to-body`, `body-x` and `body-y`.

`body-to-msg=<pole>`  
defines the horizontal coffin `<pole>` to be used for the placement of the `<animal>` beneath the `<message>`. See [interface3.pdf](#) and the documentation of `xcoffins` for information about coffin poles.

`body-x=<dimen>`  
defines a horizontal offset of `<dimen>` length of the `<animal>` from its placement beneath the `<message>`.

`body-y=<dimen>`  
defines a vertical offset of `<dimen>` length of the `<animal>` from its placement beneath the `<message>`.

`bubble=<font>`  
add `<font>` to the font definitions in use to typeset the bubble. This does not affect the `<message>` only the bubble put around it.

`bubble*=<font>`  
clear any definitions previously made (including the package default) and set the font definitions in use to typeset the bubble to `<font>`. This does not affect the `<message>` only the bubble put around it. The package default is `\verbatim@font`.

`bubble-bot-kern=<dimen>`  
specifies a vertical offset of the placement of the lower border of the bubble from the bottom of the left and right borders.

- `bubble-delim-left-1=<token list>`  
the left delimiter used if only one line of delimiters is needed. Package default is `(`.
- `bubble-delim-left-2=<token list>`  
the upper most left delimiter used if more than one line of delimiters is needed. Package default is `/`.
- `bubble-delim-left-3=<token list>`  
the left delimiters used to fill the gap if more than two lines of delimiters are needed. Package default is `|`.
- `bubble-delim-left-4=<token list>`  
the lower most left delimiter used if more than one line of delimiters is needed. Package default is `\`.
- `bubble-delim-right-1=<token list>`  
the right delimiter used if only one line of delimiters is needed. Package default is `)`.
- `bubble-delim-right-2=<token list>`  
the upper most right delimiter used if more than one line of delimiters is needed. Package default is `\`.
- `bubble-delim-right-3=<token list>`  
the right delimiters used to fill the gap if more than two lines of delimiters are needed. Package default is `|`.
- `bubble-delim-right-4=<token list>`  
the lower most right delimiter used if more than one line of delimiters is needed. Package default is `/`.
- `bubble-delim-top=<token list>`  
the delimiter used to create the top and bottom border of the bubble. The package default is `{-}` (the braces are important to suppress ligatures here).
- `bubble-side-kern=<dimen>`  
specifies the kerning used to move the sideways delimiters added to fill the gap for more than two lines of bubble height. (the left one is moved to the left, the right one to the right)
- `bubble-top-kern=<dimen>`  
specifies a vertical offset of the placement of the upper border of the bubble from the top of the left and right borders.
- `c`  
shortcut for `out-v=vc`.
- `col=<column>`  
specifies the used column specifier used for the `<message>` enclosing `tabular` for `arg=tab` and `arg=tab*`. Has precedence over `msg-align`. You can also use more than one column this way: `\ducksay[arg=tab,col=cc]{ You & can \\ do & it }` would be valid syntax.
- `hpad=<count>`  
Add `<count>` times more `bubble-delim-top` instances than necessary to the upper and lower border of the bubble. Package default is 2.

```

(-----)
 \      /
  \    / \
   \  /   \
    \|     \|
     \|     \|
    /      /
   /        /
  /          /
 /            /
/              /

```

( 10 )

- `msg-to-body=<pole>`  
 defines the horizontal coffin `<pole>` to be used as the reference point for the placement of the `<animal>` beneath the `<message>`. See [interface3.pdf](#) and the documentation of [xcoffins](#) for information about coffin poles.
- `no-bubble=<bool>`  
 If `true` the `<message>` will not be surrounded by a bubble. Package default is of course `false`.
- `none=<bool>` One could say this is a special animal. If `true` no animal body will be used (resulting in just the speech bubble). Package default is of course `false`.
- `out-h=<pole>`  
 defines the horizontal coffin `<pole>` to be used as the anchor point for the print out of the complete result of `\ducksay` and `\duckthink`. See [interface3.pdf](#) and the documentation of [xcoffins](#) for information about coffin poles.
- `out-v=<pole>`  
 defines the vertical coffin `<pole>` to be used as the anchor point for the print out of the complete result of `\ducksay` and `\duckthink`. See [interface3.pdf](#) and the documentation of [xcoffins](#) for information about coffin poles.
- `out-x=<dimen>`  
 specifies an additional horizontal offset of the print out of the complete result of `\ducksay` and `\duckthink`.
- `out-y=<dimen>`  
 specifies an additional vertical offset of the print out of the complete result of `\ducksay` and `\duckthink`.
- `strip-spaces=<bool>`  
 if set `true` leading and trailing spaces are stripped from the `<message>` if `arg=box` is used. Initially this is set to `false`.
- `t`  
 shortcut for `out-v=t`.
- `vpad=<count>`  
 add `<count>` to the lines used for the bubble, resulting in `<count>` more lines than necessary to enclose the `<message>` inside of the bubble.
- `wd=<count>`  
 specifies the width of the `<message>` to be fixed to `<count>` times the width of an upper case M in the `<message>`'s font declaration. A value smaller than 0 is considered deactivated, else the width is considered as fixed. For a fixed width the argument of `\ducksay` and `\duckthink` is read in as a `\vbox` for `arg=box` and the column definition uses a `p`-type column for `arg=tab` and `arg=tab*`. If both `wd` is not smaller than 0 and `wd*` is not smaller than 0pt, `wd*` will take precedence.
- `wd*=<dimen>`  
 specifies the width of the `<message>` to be fixed to `<dimen>`. A value smaller than 0pt is considered deactivated, else the width is considered as fixed. For a fixed width the argument of `\ducksay` and `\duckthink` is read in as a `\vbox` for `arg=box` and the column definition uses a `p`-type column for `arg=tab` and `arg=tab*`. If both `wd` is not smaller than 0 and `wd*` is not smaller than 0pt, `wd*` will take precedence.

```

( 11 )
  \ .-'\ \
   " \   \
  /    \  ( . )
 /-----\

```



I'm the  
new one.

Deus  
vult!\*

\[T]/

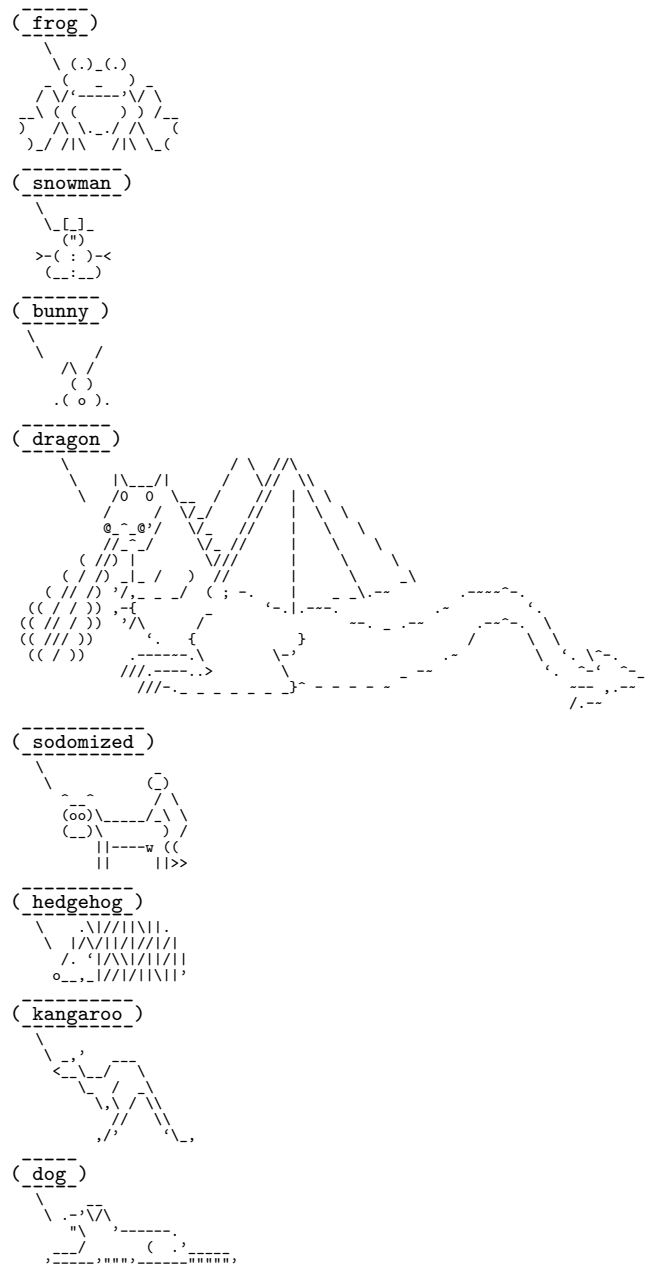
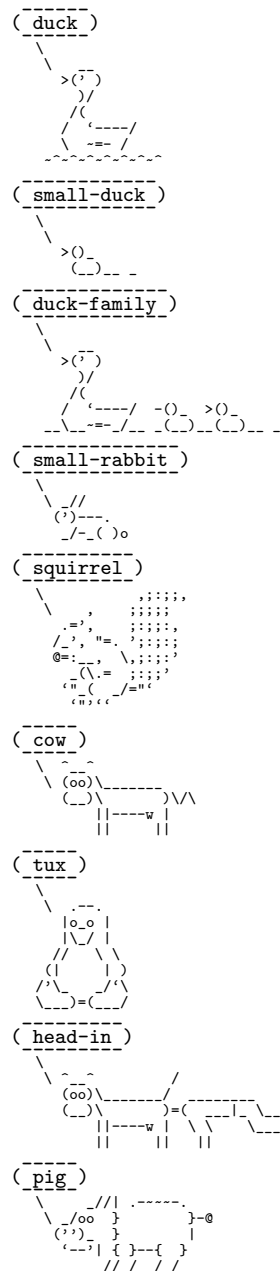


## 1.5 Dependencies

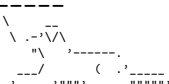
The package depends on the two packages `xparse` and `l3keys2e` and all of their dependencies. Version 2 additionally depends on `array` and `grabbbox`.

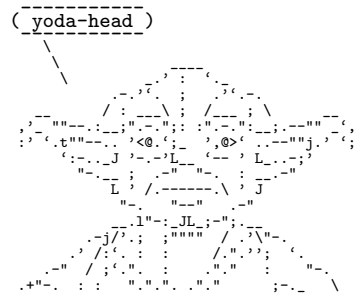
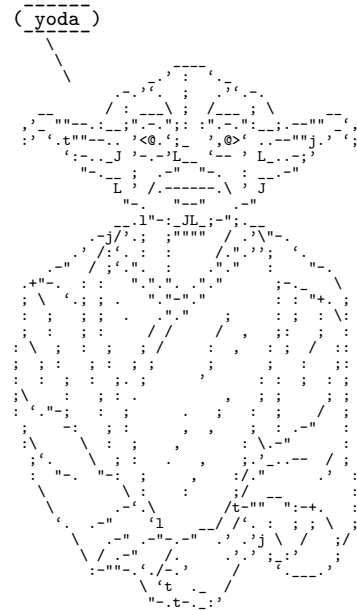
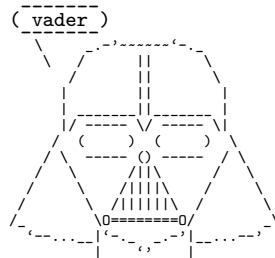
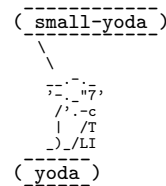
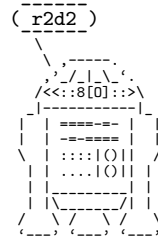
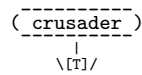
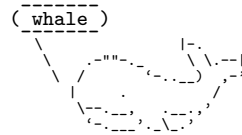
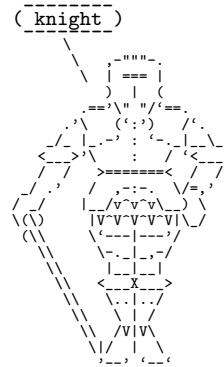
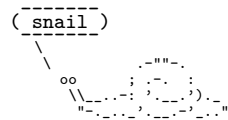
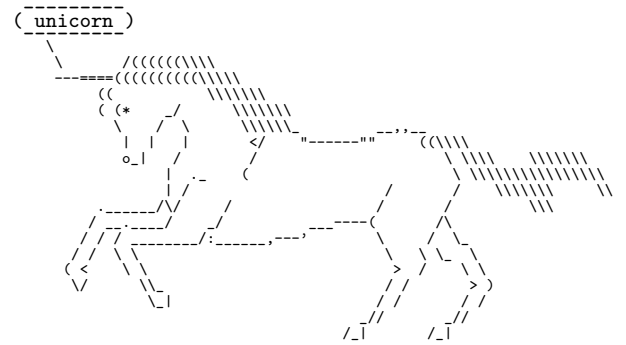
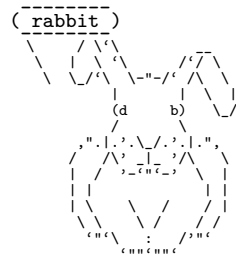
## 1.6 Available Animals

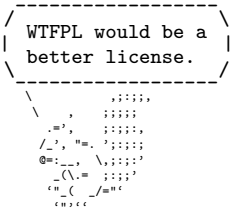
The following animals are provided by this package. I did not create them (but altered some), they belong to their original creators.



\*Latin; "I'm new, too."







1.7 Miscellaneous

This work may be distributed and/or modified under the conditions of the L<sup>A</sup>T<sub>E</sub>X Project Public License (LPPL), either version 1.3c of this license or (at your option) any later version. The latest version of this license is in the file: <http://www.latex-project.org/lppl.txt>

The package is hosted on [https://github.com/Skillmon/ltx\\_ducksay](https://github.com/Skillmon/ltx_ducksay), you might report bugs there.

Only rebel scum reads  
documentation!

Join the dark side,  
read the implementation.

 $1 \quad \langle^* \text{pkg} \rangle$ 

( 15 )



```

32 ,align .tl_set:N      = \l_ducksay_align_tl
33 ,align .value_required:n = true
34 ,wd .int_set:N        = \l_ducksay_msg_width_int
35 ,wd .initial:n        = -\c_max_int
36 ,wd .value_required:n = true
37 ,ht .int_set:N        = \l_ducksay_msg_height_int
38 ,ht .initial:n        = -\c_max_int
39 ,ht .value_required:n = true
40 ,animal .code:n       =
41   { \keys_define:nn { ducksay } { default_animal .meta:n = { #1 } } }
42 ,animal .initial:n    = duck
43 ,msg-align .tl_set:N  = \l_ducksay_msg_align_tl
44 ,msg-align .initial:n = 1
45 ,msg-align .value_required:n = true
46 ,rel-align .tl_set:N  = \l_ducksay_rel_align_tl
47 ,rel-align .initial:n = 1
48 ,rel-align .value_required:n = true
49 ,ligatures .tl_set:N  = \l_ducksay_ligatures_tl
50 ,ligatures .initial:n = { '<>','-' }
51 ,tail-1 .tl_set:N     = \l_ducksay_tail_symbol_out_one_tl
52 ,tail-1 .initial:x    = \c_backslash_str
53 ,tail-2 .tl_set:N     = \l_ducksay_tail_symbol_out_two_tl
54 ,tail-2 .initial:x    = \c_backslash_str
55 ,no-tail .meta:n      = { tail-1 = { ~ }, tail-2 = { ~ } }
56 ,think .meta:n       = { tail-1 = { 0 }, tail-2 = { o } }
57 ,say .code:n          =
58   {
59     \exp_args:Nx \DucksayOptions
60     { tail-1 = { \c_backslash_str }, tail-2 = { \c_backslash_str } }
61   }
62 ,version .choice:
63 ,version / 1 .code:n =
64   {
65     \bool_set_false:N \l_ducksay_version_two_bool
66     \bool_set_true:N  \l_ducksay_version_one_bool
67   }
68 ,version / 2 .code:n =
69   {
70     \bool_set_false:N \l_ducksay_version_one_bool
71     \bool_set_true:N  \l_ducksay_version_two_bool
72   }
73 ,version .initial:n = 2
74 ,add-think .code:n   = \msg_error:nn { ducksay } { deprecated-key }
75 }
76 \ProcessKeysOptions { ducksay }
77
78   Undefine the load-time-only keys
79 \keys_define:nn { ducksay }
80   {
81     version .code:n = \msg_error:nnn { ducksay } { load-time-only } { version }
82   }

```

#### 2.1.4.1 Keys for \AddAnimal

Define keys meant for \AddAnimal and \AddColoredAnimal only in their own regime:

```

(-----)
( 16 )
\ .-'\ \
  " \  '-----
  /   ( .'------
  /-----)-----)-----)

```

```

81 \keys_define:nn { ducksay / add-animal }
82 {
83   ,tail-symbol .code:n =
84     \tl_set:Nx \l_ducksay_tail_symbol_in_tl { \tl_to_str:n { #1 } }
85   ,tail-symbol .initial:o = \c_backslash_str
86   ,tail-count .int_set:N = \l_ducksay_tail_symbol_count_int
87   ,tail-count .initial:n = 2
88 }

```

## 2.1.5 Functions

### 2.1.5.1 Generating Variants of External Functions

```

89 \cs_generate_variant:Nn \tl_if_eq:nnT { VnT }
90 \cs_generate_variant:Nn \tl_replace_once:Nnn { NVn }
91 \cs_generate_variant:Nn \tl_replace_all:Nnn { NVn }

```

### 2.1.5.2 Internal

`\ducksay_replace_verb_newline:Nn`

```

92 \cs_new_protected:Npx \ducksay_replace_verb_newline:Nn #1 #2
93 {
94   \tl_replace_all:Nnn #1 { \char_generate:nn { 13 } { 12 } } { #2 }
95 }

```

*(End definition for \ducksay\_replace\_verb\_newline:Nn. This function is documented on page ??.)*

`\ducksay_replace_verb_newline_newline:Nn`

```

96 \cs_new_protected:Npx \ducksay_replace_verb_newline_newline:Nn #1 #2
97 {
98   \tl_replace_all:Nnn #1
99   { \char_generate:nn { 13 } { 12 } \char_generate:nn { 13 } { 12 } } { #2 }
100 }

```

*(End definition for \ducksay\_replace\_verb\_newline\_newline:Nn. This function is documented on page ??.)*

`\ducksay_process_verb_newline:nnn`

```

101 \cs_new_protected:Npn \ducksay_process_verb_newline:nnn #1 #2 #3
102 {
103   \tl_set:Nn \ProcessedArgument { #3 }
104   \ducksay_replace_verb_newline_newline:Nn \ProcessedArgument { #2 }
105   \ducksay_replace_verb_newline:Nn \ProcessedArgument { #1 }
106 }

```

*(End definition for \ducksay\_process\_verb\_newline:nnn. This function is documented on page ??.)*

`\ducksay_add_animal_inner:nnnn`

```

107 \cs_new_protected:Npn \ducksay_add_animal_inner:nnnn #1 #2 #3 #4
108 {
109   \group_begin:
110     \keys_set:nn { ducksay / add-animal } { #1 }
111     \tl_set:Nn \l_ducksay_tmpa_tl { \ #3 }
112     \int_compare:nNnTF { \l_ducksay_tail_symbol_count_int } < { \c_zero_int }
113     {
114       \tl_replace_once:NVn

```

```

( 17 )
\ .-'\
" \
----- ( .) -----
) -----) -----) -----) -----)

```

```

115         \l_ducksay_tmpa_tl
116         \l_ducksay_tail_symbol_in_tl
117         \l_ducksay_tail_symbol_out_one_tl
118     \tl_replace_all:N\N
119         \l_ducksay_tmpa_tl
120         \l_ducksay_tail_symbol_in_tl
121         \l_ducksay_tail_symbol_out_two_tl
122 }
123 {
124     \int_compare:nNnT { \l_ducksay_tail_symbol_count_int } >
125     { \c_zero_int }
126     {
127         \tl_replace_once:N\N
128             \l_ducksay_tmpa_tl
129             \l_ducksay_tail_symbol_in_tl
130             \l_ducksay_tail_symbol_out_one_tl
131         \int_step_inline:nnn { 2 } { \l_ducksay_tail_symbol_count_int }
132         {
133             \tl_replace_once:N\N
134                 \l_ducksay_tmpa_tl
135                 \l_ducksay_tail_symbol_in_tl
136                 \l_ducksay_tail_symbol_out_two_tl
137         }
138     }
139 }
140 \tl_map_inline:Nn \l_ducksay_ligatures_tl
141 { \tl_replace_all:Nnn \l_ducksay_tmpa_tl { ##1 } { { ##1 } } }
142 \ducksay_replace_verb_newline:Nn \l_ducksay_tmpa_tl
143 { \tabularnewline\null }
144 \exp_args:NNnV
145 \group_end:
146 \tl_set:cn { l_ducksay_animal_#2_tl } \l_ducksay_tmpa_tl
147 \exp_args:Nnx \keys_define:nn { ducksay }
148 {
149     #2 .code:n =
150     {
151         \exp_not:n { \tl_set_eq:NN \l_ducksay_animal_tl }
152         \exp_after:wN \exp_not:N \cs:w l_ducksay_animal_#2_tl \cs_end:
153         \exp_not:n { \exp_args:NV \DucksayOptions }
154         \exp_after:wN
155         \exp_not:N \cs:w l_ducksay_animal_#2_options_tl \cs_end:
156     }
157 }
158 \tl_if_exist:cF { l_ducksay_animal_#2_options_tl }
159 { \tl_new:c { l_ducksay_animal_#2_options_tl } }
160 \IfBooleanT { #4 }
161 { \keys_define:nn { ducksay } { default_animal .meta:n = { #2 } } }
162 }
163 \cs_generate_variant:Nn \ducksay_add_animal_inner:nnnn { nnVn }

```

(End definition for \ducksay\_add\_animal\_inner:nnnn. This function is documented on page ??.)

### 2.1.5.3 Document level

\DefaultAnimal

```

164 \NewDocumentCommand \DefaultAnimal { m }
165 {
166   \keys_define:nn { ducksay } { default_animal .meta:n = { #1 } }
167 }

```

(End definition for \DefaultAnimal. This function is documented on page 2.)

\ DucksayOptions

```

168 \NewDocumentCommand \DucksayOptions { m }
169 {
170   \keys_set:nn { ducksay } { #1 }
171 }

```

(End definition for \DucksayOptions. This function is documented on page 2.)

\AddAnimal

```

172 \NewDocumentCommand \AddAnimal { s O{} m +v }
173 {
174     \ducksay_add_animal_inner:nnnn { #2 } { #3 } { #4 } { #1 }
175 }

```

(End definition for `\AddAnimal`. This function is documented on page 3.)

\AddColoredAnimal

```

176 \NewDocumentCommand \AddColoredAnimal { s O{} m +v }
177 {
178   \tl_set:Nn \l_ducksay_tmpa_tl { #4 }
179   \regex_replace_all:NnN \c_ducksay_color_delim_regex
180     { \c{bgroup}\c{color}\cB{\1\cE}\2\c{egroup} }
181   \l_ducksay_tmpa_tl
182   \regex_replace_all:NnN \c_ducksay_color_regex
183     { \c{color}\cB{\1\cE}\ }
184   \l_ducksay_tmpa_tl
185   \regex_replace_all:NnN \c_ducksay_textcolor_regex
186     { \c{textcolor}\cB{\1\cE}\cB{\2\cE}\ }
187   \l_ducksay_tmpa_tl
188   \ducksay_add_animal_inner:nnVn { #2 } { #3 } \l_ducksay_tmpa_tl { #1 }
189 }

```

(End definition for \AddColoredAnimal. This function is documented on page 3.)

\AnimalOptions

```

190 \NewDocumentCommand \AnimalOptions { s m m }
191 {
192   \tl_if_exist:cTF { l_ducksay_animal_#2_options_tl }
193   {
194     \IfBooleanTF { #1 }
195     { \tl_set:cn }
196     { \tl_put_right:cn }
197   }
198   { \tl_set:cn }
199   { l_ducksay_animal_#2_options_tl } { #3, }
200 }

```

(End definition for \AnimalOptions. This function is documented on page 3.)

### 2.1.6 Load the Correct Version and the Animals

```
201 \bool_if:NT \l_ducksay_version_one_bool
202   { \file_input:n { ducksay.code.v1.tex } }
203 \bool_if:NT \l_ducksay_version_two_bool
204   { \file_input:n { ducksay.code.v2.tex } }

205 \ExplSyntaxOff
206 \input{ducksay.animals.tex}

207 </pkg>
```

## 2.2 Version 1

208  $\langle *code.v1 \rangle$

### 2.2.1 Functions

#### 2.2.1.1 Internal

`\ducksay_longest_line:n` Calculate the length of the longest line

```

209 \cs_new:Npn \ducksay_longest_line:n #1
210 {
211   \int_incr:N \l_ducksay_msg_height_int
212   \exp_args:NNx \tl_set:Nn \l_ducksay_tmpa_tl { #1 }
213   \regex_replace_all:nnN { \s } { \c { space } } \l_ducksay_tmpa_tl
214   \int_set:Nn \l_ducksay_msg_width_int
215   {
216     \int_max:nn
217     { \l_ducksay_msg_width_int } { \tl_count:N \l_ducksay_tmpa_tl }
218   }
219 }
```

(End definition for `\ducksay_longest_line:n`. This function is documented on page ??.)

`\ducksay_open_bubble:` Draw the opening bracket of the bubble

```

220 \cs_new:Npn \ducksay_open_bubble:
221 {
222   \begin{tabular}{@{}l@{}}
223     \null\
224     \int_compare:nNnTF { \l_ducksay_msg_height_int } = { 1 } { ( }
225     {
226       /
227       \int_step_inline:nnn
228       { 3 } { \l_ducksay_msg_height_int } { \\kern-0.2em| }
229       \\detokenize{\ }
230     }
231     \\[-1ex]\null
232   \end{tabular}
233   \begin{tabular}{@{}l@{}}
234     _\\
235     \int_step_inline:nnn { 2 } { \l_ducksay_msg_height_int } { \\ } \\[-1ex]
236     \mbox { - }
237   \end{tabular}
238 }
```

(End definition for `\ducksay_open_bubble:.` This function is documented on page ??.)

`\ducksay_close_bubble:` Draw the closing bracket of the bubble

```

239 \cs_new:Npn \ducksay_close_bubble:
240 {
241   \begin{tabular}{@{}l@{}}
242     _\\
243     \int_step_inline:nnn { 2 } { \l_ducksay_msg_height_int } { \\ } \\[-1ex]
244     { - }
245   \end{tabular}
246   \begin{tabular}{@{}r@{}}
247     \null\
```

```

( 21 )
\ .-' \
" \
_ / ( . '
) -----)

```

```

248 \int_compare:nNnTF { \l_ducksay_msg_height_int } = { 1 }
249 { ) }
250 {
251   \detokenize { \ }
252   \int_step_inline:nnn
253     { 3 } { \l_ducksay_msg_height_int } { \\\kern-0.2em }
254   \\\
255 }
256 \\\[-1ex]\null
257 \end{tabular}
258 }

```

(End definition for `\ducksay_close_bubble:.` This function is documented on page ??.)

`\ducksay_print_msg:nn` Print out the message

```

259 \cs_new:Npn \ducksay_print_msg:nn #1 #2
260 {
261   \begin{tabular}{@{} #2 @{}}
262     \int_step_inline:nn { \l_ducksay_msg_width_int } { _ } \\\
263     #1\\[-1ex]
264     \int_step_inline:nn { \l_ducksay_msg_width_int } { { - } }
265   \end{tabular}
266 }
267 \cs_generate_variant:Nn \ducksay_print_msg:nn { nV }

```

(End definition for `\ducksay_print_msg:nn`. This function is documented on page ??.)

`\ducksay_print:nn` Print out the whole thing

```

268 \cs_new:Npn \ducksay_print:nn #1 #2
269 {
270   \int_compare:nNnTF { \l_ducksay_msg_width_int } < { 0 }
271   {
272     \int_zero:N \l_ducksay_msg_height_int
273     \seq_set_split:Nnn \l_ducksay_msg_lines_seq { \\\ } { #1 }
274     \seq_map_function:NN \l_ducksay_msg_lines_seq \ducksay_longest_line:n
275   }
276   {
277     \int_compare:nNnT { \l_ducksay_msg_height_int } < { 0 }
278     {
279       \regex_count:nnN { \c { \\\ } } { #1 } \l_ducksay_msg_height_int
280       \int_incr:N \l_ducksay_msg_height_int
281     }
282   }
283   \group_begin:
284     \frenchspacing
285     \verbatim@font
286     \@noligs
287     \begin{tabular}[\l_ducksay_align_tl]{@{}#2@{}}
288       \l_ducksay_bubble_tl
289       \begin{tabular}{@{}l@{}}
290         \ducksay_open_bubble:
291         \ducksay_print_msg:nV { #1 } \l_ducksay_msg_align_tl
292         \ducksay_close_bubble:
293       \end{tabular}\\\
294       \l_ducksay_body_tl

```

```

295         \begin{tabular}{@{}l@{}}
296             \l_ducksay_animal_tl
297         \end{tabular}
298     \end{tabular}
299     \group_end:
300 }
301 \cs_generate_variant:Nn \ducksay_print:nn { nV }

```

(End definition for `\ducksay_print:nn`. This function is documented on page ??.)

```

\ducksay_say_and_think:nn Reset some variables
302 \cs_new:Npn \ducksay_say_and_think:nn #1 #2
303 {
304     \group_begin:
305     \int_set:Nn \l_ducksay_msg_width_int { -\c_max_int }
306     \int_set:Nn \l_ducksay_msg_height_int { -\c_max_int }
307     \keys_set:nn { ducksay } { #1 }
308     \tl_if_empty:NT \l_ducksay_animal_tl
309         { \keys_set:nn { ducksay } { default_animal } }
310     \ducksay_print:nV { #2 } \l_ducksay_rel_align_tl
311     \group_end:
312 }

```

(End definition for `\ducksay_say_and_think:nn`. This function is documented on page ??.)

### 2.2.1.2 Document level

#### `\ducksay`

```

313 \NewDocumentCommand \ducksay { 0{} m }
314 {
315     \ducksay_say_and_think:nn { #1 } { #2 }
316 }

```

(End definition for `\ducksay`. This function is documented on page 7.)

#### `\duckthink`

```

317 \NewDocumentCommand \duckthink { 0{} m }
318 {
319     \ducksay_say_and_think:nn { think, #1 } { #2 }
320 }

```

(End definition for `\duckthink`. This function is documented on page 7.)

```

321 \</code.v1>

```



## 2.3 Version 2

322 `<*code.v2>`

Load the additional dependencies of version 2.

323 `\RequirePackage{array,grabbox}`

### 2.3.1 Messages

324 `\msg_new:nnn { ducksay } { justify-unavailable }`

325 `{`

326 `Justified-content-is-not-available-for-tabular-argument-mode-without-fixed-`  
327 `width.~'l'~column-is-used-instead.`

328 `}`

329 `\msg_new:nnn { ducksay } { unknown-message-alignment }`

330 `{`

331 `The-specified-message-alignment~'\exp_not:n { #1 }'~is-unknown.~`  
332 `'l'~is-used-as-fallback.`

333 `}`

334 `\msg_new:nnn { ducksay } { v1-key-only }`

335 `{ The~'\l_keys_key_tl'~key-is-only-available-for~'version=1'. }`

### 2.3.2 Variables

#### 2.3.2.1 Token Lists

336 `\tl_new:N \l_ducksay_msg_align_vbox_tl`

#### 2.3.2.2 Boxes

337 `\box_new:N \l_ducksay_msg_box`

#### 2.3.2.3 Bools

338 `\bool_new:N \l_ducksay_eat_arg_box_bool`

339 `\bool_new:N \l_ducksay_eat_arg_tab_verb_bool`

340 `\bool_new:N \l_ducksay_mirrored_body_bool`

#### 2.3.2.4 Coffins

341 `\coffin_new:N \l_ducksay_body_coffin`

342 `\coffin_new:N \l_ducksay_bubble_close_coffin`

343 `\coffin_new:N \l_ducksay_bubble_open_coffin`

344 `\coffin_new:N \l_ducksay_bubble_top_coffin`

345 `\coffin_new:N \l_ducksay_msg_coffin`

#### 2.3.2.5 Dimensions

346 `\dim_new:N \l_ducksay_hpad_dim`

347 `\dim_new:N \l_ducksay_bubble_bottom_kern_dim`

348 `\dim_new:N \l_ducksay_bubble_top_kern_dim`

349 `\dim_new:N \l_ducksay_msg_width_dim`

### 2.3.3 Options

350 `\keys_define:nn { ducksay }`

351 `{`

352 `,arg .choice:`

353 `,arg / box .code:n = \bool_set_true:N \l_ducksay_eat_arg_box_bool`

354 `,arg / tab .code:n =`

355 `{`

356 `\bool_set_false:N \l_ducksay_eat_arg_box_bool`

357 `\bool_set_false:N \l_ducksay_eat_arg_tab_verb_bool`

( 24 )  
-----  
\\ .-'\V\\  
"\\  
----- ( . )-----  
)-----)-----)-----)

```

358     }
359 ,arg / tab* .code:n =
360     {
361         \bool_set_false:N \l_ducksay_eat_arg_box_bool
362         \bool_set_true:N  \l_ducksay_eat_arg_tab_verb_bool
363     }
364 ,arg .initial:n = tab
365 ,wd* .dim_set:N = \l_ducksay_msg_width_dim
366 ,wd* .initial:n = -\c_max_dim
367 ,wd* .value_required:n = true
368 ,none .bool_set:N = \l_ducksay_no_body_bool
369 ,no-bubble .bool_set:N = \l_ducksay_no_bubble_bool
370 ,body-mirrored .bool_set:N = \l_ducksay_mirrored_body_bool
371 ,ignore-body .bool_set:N = \l_ducksay_ignored_body_bool
372 ,body-x .dim_set:N = \l_ducksay_body_x_offset_dim
373 ,body-x .value_required:n = true
374 ,body-y .dim_set:N = \l_ducksay_body_y_offset_dim
375 ,body-y .value_required:n = true
376 ,body-to-msg .tl_set:N = \l_ducksay_body_to_msg_align_body_tl
377 ,msg-to-body .tl_set:N = \l_ducksay_body_to_msg_align_msg_tl
378 ,body-align .choice:
379 ,body-align / l .meta:n = { body-to-msg = l , msg-to-body = l }
380 ,body-align / c .meta:n = { body-to-msg = hc , msg-to-body = hc }
381 ,body-align / r .meta:n = { body-to-msg = r , msg-to-body = r }
382 ,body-align .initial:n = l
383 ,msg-align .choice:
384 ,msg-align / l .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { l } }
385 ,msg-align / c .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { c } }
386 ,msg-align / r .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { r } }
387 ,msg-align / j .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { j } }
388 ,msg-align-l .tl_set:N = \l_ducksay_msg_align_l_tl
389 ,msg-align-l .initial:n = \raggedright
390 ,msg-align-c .tl_set:N = \l_ducksay_msg_align_c_tl
391 ,msg-align-c .initial:n = \centering
392 ,msg-align-r .tl_set:N = \l_ducksay_msg_align_r_tl
393 ,msg-align-r .initial:n = \raggedleft
394 ,msg-align-j .tl_set:N = \l_ducksay_msg_align_j_tl
395 ,msg-align-j .initial:n = {}
396 ,out-h .tl_set:N = \l_ducksay_output_h_pole_tl
397 ,out-h .initial:n = l
398 ,out-v .tl_set:N = \l_ducksay_output_v_pole_tl
399 ,out-v .initial:n = vc
400 ,out-x .dim_set:N = \l_ducksay_output_x_offset_dim
401 ,out-x .value_required:n = true
402 ,out-y .dim_set:N = \l_ducksay_output_y_offset_dim
403 ,out-y .value_required:n = true
404 ,t .meta:n = { out-v = t }
405 ,c .meta:n = { out-v = vc }
406 ,b .meta:n = { out-v = b }
407 ,body* .tl_set:N = \l_ducksay_body_fount_tl
408 ,msg* .tl_set:N = \l_ducksay_msg_fount_tl
409 ,bubble* .tl_set:N = \l_ducksay_bubble_fount_tl
410 ,body* .initial:n = \verbatim@font
411 ,msg* .initial:n = \verbatim@font

```

```

412 ,bubble* .initial:n = \verbatim@font
413 ,body .code:n = \tl_put_right:Nn \l_ducksay_body_fount_tl { #1 }
414 ,msg .code:n = \tl_put_right:Nn \l_ducksay_msg_fount_tl { #1 }
415 ,bubble .code:n = \tl_put_right:Nn \l_ducksay_bubble_fount_tl { #1 }
416 ,MSG .meta:n = { msg = #1 , bubble = #1 }
417 ,MSG* .meta:n = { msg* = #1 , bubble* = #1 }
418 ,hpad .int_set:N = \l_ducksay_hpad_int
419 ,hpad .initial:n = 2
420 ,hpad .value_required:n = true
421 ,vpad .int_set:N = \l_ducksay_vpad_int
422 ,vpad .value_required:n = true
423 ,col .tl_set:N = \l_ducksay_msg_tabular_column_tl
424 ,bubble-top-kern .tl_set:N = \l_ducksay_bubble_top_kern_tl
425 ,bubble-top-kern .initial:n = { -.5ex }
426 ,bubble-top-kern .value_required:n = true
427 ,bubble-bot-kern .tl_set:N = \l_ducksay_bubble_bottom_kern_tl
428 ,bubble-bot-kern .initial:n = { .2ex }
429 ,bubble-bot-kern .value_required:n = true
430 ,bubble-side-kern .tl_set:N = \l_ducksay_bubble_side_kern_tl
431 ,bubble-side-kern .initial:n = { .2em }
432 ,bubble-side-kern .value_required:n = true
433 ,bubble-delim-top .tl_set:N = \l_ducksay_bubble_delim_top_tl
434 ,bubble-delim-left-1 .tl_set:N = \l_ducksay_bubble_delim_left_a_tl
435 ,bubble-delim-left-2 .tl_set:N = \l_ducksay_bubble_delim_left_b_tl
436 ,bubble-delim-left-3 .tl_set:N = \l_ducksay_bubble_delim_left_c_tl
437 ,bubble-delim-left-4 .tl_set:N = \l_ducksay_bubble_delim_left_d_tl
438 ,bubble-delim-right-1 .tl_set:N = \l_ducksay_bubble_delim_right_a_tl
439 ,bubble-delim-right-2 .tl_set:N = \l_ducksay_bubble_delim_right_b_tl
440 ,bubble-delim-right-3 .tl_set:N = \l_ducksay_bubble_delim_right_c_tl
441 ,bubble-delim-right-4 .tl_set:N = \l_ducksay_bubble_delim_right_d_tl
442 ,bubble-delim-top .initial:n = { { - } }
443 ,bubble-delim-left-1 .initial:n = (
444 ,bubble-delim-left-2 .initial:n = /
445 ,bubble-delim-left-3 .initial:n = |
446 ,bubble-delim-left-4 .initial:n = \c_backslash_str
447 ,bubble-delim-right-1 .initial:n = )
448 ,bubble-delim-right-2 .initial:n = \c_backslash_str
449 ,bubble-delim-right-3 .initial:n = |
450 ,bubble-delim-right-4 .initial:n = /
451 ,strip-spaces .bool_set:N = \l_ducksay_msg_strip_spaces_bool
452 }

```

Redefine keys only intended for version 1 to throw an error:

```

453 \clist_map_inline:nn
454   { align, rel-align }
455   {
456     \keys_define:nn { ducksay }
457       { ##1 .code:n = \msg_error:nn { ducksay } { v1-key-only } }
458   }

```

### 2.3.4 Functions

#### 2.3.4.1 Internal

evaluate message alignment fixed width common:

```
459 \cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_common:
```

( 26 )

```

460 {
461   \str_case:Vn \l_ducksay_msg_align_tl
462   {
463     { l } { \exp_not:N \l_ducksay_msg_align_l_tl }
464     { c } { \exp_not:N \l_ducksay_msg_align_c_tl }
465     { r } { \exp_not:N \l_ducksay_msg_align_r_tl }
466     { j } { \exp_not:N \l_ducksay_msg_align_j_tl }
467   }
468 }

```

(End definition for \ducksay\_evaluate\_message\_alignment\_fixed\_width\_common:. This function is documented on page ??.)

luate\_message\_alignment\_fixed\_width\_tabular:

```

469 \cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_tabular:
470 {
471   \tl_if_empty:NT \l_ducksay_msg_tabular_column_tl
472   {
473     \tl_set:Nx \l_ducksay_msg_tabular_column_tl
474     {
475       >
476       {
477         \ducksay_evaluate_message_alignment_fixed_width_common:
478         \exp_not:N \arraybackslash
479       }
480       p { \exp_not:N \l_ducksay_msg_width_dim }
481     }
482   }
483 }

```

(End definition for \ducksay\_evaluate\_message\_alignment\_fixed\_width\_tabular:. This function is documented on page ??.)

evaluate\_message\_alignment\_fixed\_width\_vbox:

```

484 \cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_vbox:
485 {
486   \tl_set:Nx \l_ducksay_msg_align_vbox_tl
487   { \ducksay_evaluate_message_alignment_fixed_width_common: }
488 }

```

(End definition for \ducksay\_evaluate\_message\_alignment\_fixed\_width\_vbox:. This function is documented on page ??.)

\ducksay\_calculate\_msg\_width\_from\_int:

```

489 \cs_new:Npn \ducksay_calculate_msg_width_from_int:
490 {
491   \hbox_set:Nn \l_ducksay_tmpa_box { \l_ducksay_msg_fount_tl M }
492   \dim_set:Nn \l_ducksay_msg_width_dim
493   { \l_ducksay_msg_width_int \box_wd:N \l_ducksay_tmpa_box }
494 }

```

(End definition for \ducksay\_calculate\_msg\_width\_from\_int:. This function is documented on page ??.)

\ducksay\_msg\_tabular\_begin:

```

495 \cs_new:Npn \ducksay_msg_tabular_begin:
496 {
497   \ducksay_msg_tabular_begin_inner:V \l_ducksay_msg_tabular_column_tl
498 }
499 \cs_new:Npn \ducksay_msg_tabular_begin_inner:n #1
500 {
501   \begin { tabular } { @{} #1 @{} }
502 }
503 \cs_generate_variant:Nn \ducksay_msg_tabular_begin_inner:n { V }

```

(End definition for \ducksay\_msg\_tabular\_begin:. This function is documented on page ??.)

\ducksay\_msg\_tabular\_end:

```

504 \cs_new:Npn \ducksay_msg_tabular_end:
505 {
506   \end { tabular }
507 }

```

(End definition for \ducksay\_msg\_tabular\_end:. This function is documented on page ??.)

\ducksay\_digest\_options:n

```

508 \cs_new:Npn \ducksay_digest_options:n #1
509 {
510   \group_begin:
511   \keys_set:nn { ducksay } { #1 }
512   \tl_if_empty:NT \l_ducksay_animal_tl
513   { \keys_set:nn { ducksay } { default_animal } }
514   \bool_if:NTF \l_ducksay_eat_arg_box_bool
515   {
516     \dim_compare:nNnTF { \l_ducksay_msg_width_dim } < { \c_zero_dim }
517     {
518       \int_compare:nNnTF { \l_ducksay_msg_width_int } < { \c_zero_int }
519       {
520         \cs_set_eq:NN
521         \ducksay_eat_argument:w \ducksay_eat_argument_hbox:w
522       }
523       {
524         \cs_set_eq:NN
525         \ducksay_eat_argument:w \ducksay_eat_argument_vbox:w
526         \ducksay_calculate_msg_width_from_int:
527       }
528     }
529     {
530       \cs_set_eq:NN \ducksay_eat_argument:w \ducksay_eat_argument_vbox:w
531     }
532   }
533   {
534     \dim_compare:nNnTF { \l_ducksay_msg_width_dim } < { \c_zero_dim }
535     {
536       \int_compare:nNnTF { \l_ducksay_msg_width_int } < { \c_zero_int }
537       {
538         \tl_if_empty:NT \l_ducksay_msg_tabular_column_tl
539         {

```

```

540 \str_case:Nn \l_ducksay_msg_align_tl
541 {
542   { l }
543   { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { l } }
544   { c }
545   { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { c } }
546   { r }
547   { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { r } }
548   { j } {
549     \msg_error:nn { ducksay } { justify~unavailable }
550     \tl_set:Nn \l_ducksay_msg_tabular_column_tl { l }
551   }
552 }
553 }
554 {
555   \ducksay_calculate_msg_width_from_int:
556   \ducksay_evaluate_message_alignment_fixed_width_tabular:
557 }
558 }
559 {
560   \ducksay_evaluate_message_alignment_fixed_width_tabular:
561 }
562 \cs_set_eq:NN \ducksay_eat_argument:w \ducksay_eat_argument_tabular:w
563 }
564 \ducksay_eat_argument:w
565 }

```

(End definition for \ducksay\_digest\_options:n. This function is documented on page ??.)

\ducksay\_set\_bubble\_top\_kern:

```

567 \cs_new:Npn \ducksay_set_bubble_top_kern:
568 {
569   \group_begin:
570   \l_ducksay_bubble_fount_tl
571   \exp_args:NNNx
572   \group_end:
573   \dim_set:Nn \l_ducksay_bubble_top_kern_dim
574     { \dim_eval:n { \l_ducksay_bubble_top_kern_tl } }
575 }

```

(End definition for \ducksay\_set\_bubble\_top\_kern:. This function is documented on page ??.)

```
\duksay_set_bubble_bottom_kern:
```

```

576 \cs_new:Npn \ducksay_set_bubble_bottom_kern:
577 {
578   \group_begin:
579   \l_ducksay_bubble_fount_tl
580   \exp_args:NNNx
581   \group_end:
582   \dim_set:Nn \l_ducksay_bubble_bottom_kern_dim
583     { \dim_eval:n { \l_ducksay_bubble_bottom_kern_tl } }
584 }

```

(End definition for \ducksay set bubble bottom kern:. This function is documented on page ??.)

\ducksay\_shipout:

```

585 \cs_new_protected:Npn \ducksay_shipout:
586 {
587   \hcoffin_set:Nn \l_ducksay_msg_coffin { \box_use:N \l_ducksay_msg_box }
588   \bool_if:NF \l_ducksay_no_bubble_bool
589   {
590     \hbox_set:Nn \l_ducksay_tmpa_box
591     { \l_ducksay_bubble_fount_tl \l_ducksay_bubble_delim_top_tl }
592     \int_set:Nn \l_ducksay_msg_width_int
593     {
594       \fp_eval:n
595       {
596         ceil
597         (
598           \box_wd:N \l_ducksay_msg_box / \box_wd:N \l_ducksay_tmpa_box
599         )
600       }
601     }
602     \group_begin:
603     \l_ducksay_bubble_fount_tl
604     \exp_args:NNNx
605     \group_end:
606     \int_set:Nn \l_ducksay_msg_height_int
607     {
608       \int_max:nn
609       {
610         \fp_eval:n
611         {
612           ceil
613           (
614             (
615               \box_ht:N \l_ducksay_msg_box
616               + \box_dp:N \l_ducksay_msg_box
617             )
618             / ( \arraystretch * \baselineskip )
619           )
620         }
621         + \l_ducksay_vpad_int
622       }
623       { \l_ducksay_msg_height_int }
624     }
625     \hcoffin_set:Nn \l_ducksay_bubble_open_coffin
626     {
627       \l_ducksay_bubble_fount_tl
628       \begin{tabular}{@{}l@{}}
629         \int_compare:nNnTF { \l_ducksay_msg_height_int } = { \c_one_int }
630         {
631           \l_ducksay_bubble_delim_left_a_tl
632         }
633         {
634           \l_ducksay_bubble_delim_left_b_tl\\
635           \int_step_inline:nnn
636           { 3 } { \l_ducksay_msg_height_int }
637           {

```

```

( 30 )
\ .-'\ /
" \
----- ( .) -----
)-----) H H H )-----) H H H H H )

```

```

638         \kern-\l_ducksay_bubble_side_kern_tl
639         \l_ducksay_bubble_delim_left_c_tl
640         \\
641     }
642     \l_ducksay_bubble_delim_left_d_tl
643 }
644 \end{tabular}
645 }
646 \hcoffin_set:Nn \l_ducksay_bubble_close_coffin
647 {
648     \l_ducksay_bubble_fount_tl
649     \begin{tabular}{@{}r@{}}
650         \int_compare:nNnTF { \l_ducksay_msg_height_int } = { \c_one_int }
651         {
652             \l_ducksay_bubble_delim_right_a_tl
653         }
654         {
655             \l_ducksay_bubble_delim_right_b_tl \\
656             \int_step_inline:nnn
657             { 3 } { \l_ducksay_msg_height_int }
658             {
659                 \l_ducksay_bubble_delim_right_c_tl
660                 \kern-\l_ducksay_bubble_side_kern_tl
661                 \\
662             }
663             \l_ducksay_bubble_delim_right_d_tl
664         }
665     \end{tabular}
666 }
667 \hcoffin_set:Nn \l_ducksay_bubble_top_coffin
668 {
669     \l_ducksay_bubble_fount_tl
670     \int_step_inline:nn
671     { \l_ducksay_msg_width_int + \l_ducksay_hpad_int }
672     { \l_ducksay_bubble_delim_top_tl }
673 }
674 \dim_set:Nn \l_ducksay_hpad_dim
675 {
676     (
677         \coffin_wd:N \l_ducksay_bubble_top_coffin
678         - \coffin_wd:N \l_ducksay_msg_coffin
679     ) / 2
680 }
681 \coffin_join:NnnNnnnn
682     \l_ducksay_msg_coffin          { l } { vc }
683     \l_ducksay_bubble_open_coffin { r } { vc }
684     { - \l_ducksay_hpad_dim } { \c_zero_dim }
685 \coffin_join:NnnNnnnn
686     \l_ducksay_msg_coffin          { r } { vc }
687     \l_ducksay_bubble_close_coffin { l } { vc }
688     { \l_ducksay_hpad_dim } { \c_zero_dim }
689 \ducksay_set_bubble_top_kern:
690 \ducksay_set_bubble_bottom_kern:
691 \coffin_join:NnnNnnnn

```



```

692         \l_ducksay_msg_coffin          { hc } { t }
693         \l_ducksay_bubble_top_coffin { hc } { b }
694         { \c_zero_dim } { \l_ducksay_bubble_top_kern_dim }
695     \coffin_join:NnnNnnnn
696         \l_ducksay_msg_coffin          { hc } { b }
697         \l_ducksay_bubble_top_coffin { hc } { t }
698         { \c_zero_dim } { \l_ducksay_bubble_bottom_kern_dim }
699     }
700 \bool_if:NF \l_ducksay_no_body_bool
701 {
702     \hcoffin_set:Nn \l_ducksay_body_coffin
703     {
704         \frenchspacing
705         \l_ducksay_body_fount_tl
706         \begin{tabular} { @{} l @{} }
707             \l_ducksay_animal_tl
708         \end{tabular}
709     }
710 \bool_if:NT \l_ducksay_mirrored_body_bool
711 {
712     \coffin_scale:Nnn \l_ducksay_body_coffin
713     { -\c_one_int } { \c_one_int }
714     \str_case:Vn \l_ducksay_body_to_msg_align_body_tl
715     {
716         { l } { \tl_set:Nn \l_ducksay_body_to_msg_align_body_tl { r } }
717         { r } { \tl_set:Nn \l_ducksay_body_to_msg_align_body_tl { l } }
718     }
719 }
720 \bool_if:NTF \l_ducksay_ignored_body_bool
721 { \coffin_attach:NVnNVnnn }
722 { \coffin_join:NVnNVnnn }
723 \l_ducksay_msg_coffin \l_ducksay_body_to_msg_align_msg_tl { b }
724 \l_ducksay_body_coffin \l_ducksay_body_to_msg_align_body_tl { t }
725 { \l_ducksay_body_x_offset_dim } { \l_ducksay_body_y_offset_dim }
726 }
727 \coffin_typeset:NVVnn \l_ducksay_msg_coffin
728 \l_ducksay_output_h_pole_tl \l_ducksay_output_v_pole_tl
729 { \l_ducksay_output_x_offset_dim } { \l_ducksay_output_y_offset_dim }
730 \group_end:
731 }

```

(End definition for `\ducksay_shipout:`. This function is documented on page ??.)

**2.3.4.1.1 Message Reading Functions** Version 2 has different ways of reading the message argument of `\ducksay` and `\duckthink`. They all should allow almost arbitrary content and the height and width are set based on the dimensions.

`\ducksay_eat_argument_tabular:w`

```

732 \cs_new:Npn \ducksay_eat_argument_tabular:w
733 {
734     \bool_if:NTF \l_ducksay_eat_arg_tab_verb_bool
735     { \ducksay_eat_argument_tabular_verb:w }
736     { \ducksay_eat_argument_tabular_normal:w }
737 }

```

```

( 32 )
\ .-'\
" \
----- ( .) -----
) -----) -----) -----) -----)

```

(End definition for \ducksay\_eat\_argument\_tabular:w. This function is documented on page ??.)

\ducksay\_eat\_argument\_tabular\_inner:w

```

738 \cs_new:Npn \ducksay_eat_argument_tabular_inner:w #1
739 {
740   \hbox_set:Nn \l_ducksay_msg_box
741   {
742     \l_ducksay_msg_fount_tl
743     \ducksay_msg_tabular_begin:
744     #1
745     \ducksay_msg_tabular_end:
746   }
747   \ducksay_shipout:
748 }

```

(End definition for \ducksay\_eat\_argument\_tabular\_inner:w. This function is documented on page ??.)

\ducksay\_eat\_argument\_tabular\_verb:w

```

749 \NewDocumentCommand \ducksay_eat_argument_tabular_verb:w
750 { >{ \ducksay_process_verb_newline:nnn { ~ } { ~ \par } } +v }
751 {
752   \ducksay_eat_argument_tabular_inner:w
753   {
754     \group_begin:
755     \tex_everyeof:D { \exp_not:N }
756     \exp_after:wN
757     \group_end:
758     \tex_scantokens:D { #1 }
759   }
760 }

```

(End definition for \ducksay\_eat\_argument\_tabular\_verb:w. This function is documented on page ??.)

\ducksay\_eat\_argument\_tabular\_normal:w

```

761 \NewDocumentCommand \ducksay_eat_argument_tabular_normal:w { +m }
762 { \ducksay_eat_argument_tabular_inner:w { #1 } }

```

(End definition for \ducksay\_eat\_argument\_tabular\_normal:w. This function is documented on page ??.)

\ducksay\_eat\_argument\_hbox:w

```

763 \cs_new_protected_nopar:Npn \ducksay_eat_argument_hbox:w
764 {
765   \bool_if:NTF \l_ducksay_msg_strip_spaces_bool
766   { \grabbox }
767   { \grabbox* }
768   \l_ducksay_msg_box [ \l_ducksay_msg_fount_tl ] \hbox \ducksay_shipout:
769 }

```

(End definition for \ducksay\_eat\_argument\_hbox:w. This function is documented on page ??.)

`\ducksay_eat_argument_vbox:w`

```

770 \cs_new_protected_nopar:Npn \ducksay_eat_argument_vbox:w
771 {
772   \ducksay_evaluate_message_alignment_fixed_width_vbox:
773   \bool_if:NTF \l_ducksay_msg_strip_spaces_bool
774     { \grabbox }
775     { \grabbox* }
776   [
777     \hsize \l_ducksay_msg_width_dim
778     \linewidth \hsize
779     \l_ducksay_msg_fount_tl
780     \l_ducksay_msg_align_vbox_tl
781     \@afterindentfalse
782     \@afterheading
783   ]
784   \l_ducksay_msg_box
785   \vbox \ducksay_shipout:
786 }

```

(End definition for `\ducksay_eat_argument_vbox:w`. This function is documented on page ??.)

#### 2.3.4.1.2 Generating Variants of External Functions

```

787 \cs_generate_variant:Nn \coffin_join:NnnNnnnn { NVnNVnnn }
788 \cs_generate_variant:Nn \coffin_attach:NnnNnnnn { NVnNVnnn }
789 \cs_generate_variant:Nn \coffin_typeset:Nnnnn { NVVnn }
790 \cs_generate_variant:Nn \tl_if_eq:nnT { VnT }
791 \cs_generate_variant:Nn \str_case:nn { Vn }
792 \cs_generate_variant:Nn \regex_replace_all:NnN { Nnc }

```

#### 2.3.4.2 Document level

`\ducksay`

```

793 \NewDocumentCommand \ducksay { 0{ } }
794 {
795   \ducksay_digest_options:n { #1 }
796 }

```

(End definition for `\ducksay`. This function is documented on page 7.)

`\duckthink`

```

797 \NewDocumentCommand \duckthink { 0{ } }
798 {
799   \ducksay_digest_options:n { think, #1 }
800 }

```

(End definition for `\duckthink`. This function is documented on page 7.)

801 `</code.v2>`

## 2.4 Definition of the Animals

```

802 <*animals>
803 %^A some of the below are from http://ascii.co.uk/art/kangaroo
804 \AddAnimal{duck}%>>>
805 { \
806     \
807         >(' ')
808         )/
809         /(
810         / '----/
811         \ ~=- /
812         ~~~~~~}%<<<
813 \AddAnimal{small-duck}%>>>
814 { \
815     \
816         >()_
817         (__)__ _}%<<<
818 \AddAnimal{duck-family}%>>>
819 { \
820     \
821         >(' ')
822         )/
823         /(
824         / '----/ -()_ >()_
825         __\__~=-/_ __ (__)__ (__)__ _}%<<<
826 \AddAnimal{cow}%>>>
827 { \ ^__^
828     \ (oo)\_______
829         (__)\       )\/\
830         ||----w |
831         ||     ||}%<<<
832 \AddAnimal{head-in}%>>>
833 { \
834     \ ^__^
835         (oo)\_______/
836         (__)\       )=(  ___|_ \____
837         ||----w | \ \ \ \____|
838         ||     ||     ||     ||}%<<<
839 \AddAnimal{sodomized}%>>>
840 { \
841     \      ^__^
842         (oo)\_______/ \
843         (__)\       ) /
844         ||----w ((
845         ||     ||>>}%<<<
846 \AddAnimal{tux}%>>>
847 { \
848     \ .--.
849         |o_o |
850         |/_/ |
851         // \ \
852         (|   | )

```



```

908      /\ /
909      ( )
910      .( o ).}%<<<
911 \AddAnimal{small-rabbit}%>>>
912 { \
913   \ _//
914   (')---.
915   _/_ ( )o}%<<<
916 \AddAnimal[tail-symbol=s,tail-count=3]{dragon}%>>>
917 { s      / \ //\
918   s      | \_ _/ | / \ // \ \
919   s      / 0 0 \_ _ / // | \ \
920           / \_ / \_ // | \ \
921           @_~_@' / \_ // | \ \
922           //~_// \_ // | \ \
923           ( // ) | \_ // | \ \
924           ( / / ) _ | / ) // | \ \
925           ( // / ) ' / _ _ / ( ; - . | \ \
926           (( / / ) ) , - { ' . | . - - . | \ \
927           (( // / ) ) ' \ / \_ _ _ _ _ | \ \
928           (( // / ) ) ' . { } / \_ _ _ _ _ | \ \
929           (( / ) ) . - - - - . \_ _ _ _ _ | \ \
930           /// . - - - - . > \_ _ _ _ _ | \ \
931           /// - . - - - - - } ^ - - - - - | \ \
932                                     / . - ~ } %<<<
933 %^~A http://www.ascii-art.de/ascii/def/dogs.txt
934 \AddAnimal{dog}%>>>
935 { \
936   \ .-' \ \
937   " \ '-----.
938   _ _ / ( . '-----
939   ,-----, """,-----, """,-----, """,-----}%<<<
940 %^~A http://ascii.co.uk/art/squirrel
941 \AddAnimal{squirrel}%>>>
942 { \      , ; ; ; ,
943   \      , ; ; ; ,
944   .=' , ; ; ; ,
945   /_ ' , "= . ' ; ; ; ,
946   @=: _ _ , \ , ; ; ; '
947   _ ( \ . = ; ; ; '
948   ' " _ ( _ / = " '
949   ' " , ' , ' } %<<<
950 \AddAnimal{snail}%>>>
951 { \
952   \      . - " " - .
953   oo      ; . - . :
954   \ \ _ _ . - : ' . _ _ . ' ) . _
955   " - . _ _ . ' . _ _ . - ' . _ _ . " } %<<<
956 %^~A http://www.ascii-art.de/ascii/uvw/unicorn.txt
957 \AddAnimal{unicorn}%>>>
958 { \
959   \      / ( ( ( ( ( \ \ \ \
960   ---====( ( ( ( ( ( ( \ \ \ \
961           ( ( \ \ \ \ \ \

```

```

962      ( (*      _/      \\\\\\\
963      \      / \      \\\\\\\_
964      | | |      </      "-----"      ((\\
965      o_|      /      /      \\\\\\\
966      |      .-      (      \\\\\\\
967      |      /      /      \\\\\\\
968      .-----/ \      /      \\\\\\\
969      /      /      /      /      \\\\\\\
970      /      /      /      /      \\\\\\\
971      /      /      /      /      \\\\\\\
972      ( <      \      \      >      /      \      \
973      \      \      \      \      /      /      /
974      \      \      \      \      /      /      /
975      _//      _//
976      /_ |      /_ |}%<<<
977 %^A https://asciiart.website//index.php?art=animals/other%20(water)
978 \AddAnimal[tail-count=3,tail-symbol=s]{whale}%>>>
979 { s      |-.
980 s      .-"-.      \ \---|
981 s      /      '---)      ,-'
982 |      .      /
983 \---.---,      .---,
984 '---.---,      \_.'}%<<<
985 %^A from http://www.ascii-art.de/ascii/s/starwars.txt :
986 \AddAnimal[tail-count=3]{yoda}%>>>
987 {
988 \
989      '---
990      .-' ;      '---
991      / : ---\ ;      /--- ; \
992      ,-' "----:---;"---:"---:---;"---"---'
993      : 't"---.' <@.' ;      ,@' '---"j.' ' ;
994      '---J '---'L--- '---' L---;'
995      "-.--- ;      "-.--- : ---"
996      L ' /-----.\ ' J
997      "-.--- "---" .-"
998      ---.l"---:JL_;"---;---
999      .-j/' ;      ;"---" / .' \---
1000      .' /:' : :      /.".' ;      '
1001      .-" / ;      " :      " :      "-.
1002      .+"- : :      " " " " " ;--- \
1003      ; \ ' ;      " " " " : : "+" ;
1004      : ;      ;      " " " ;      ; \
1005      ; :      ;      / /      / , ;      ;
1006      : \ ;      ;      / : , : ;      / :
1007      ; ;      ;      ;      ;      ; ;
1008      : ;      ;      ;      ;      ; ;
1009      ; \ : ;      ;      , ;      ; ;
1010      : ' " ;      ;      ;      / ;
1011      ;      - : ;      , ;      "- :
1012      : \      \ : ;      , : \ "- :
1013      ;      ' \ : ;      , ;      '--- / ;
1014      :      "-.--- : ;      , : / " ' :
1015      \      \ :      ;      / --- :

```

```

1016 \      .-'\      /t-" " :--+ . :
1017 ' . -" '1 --/ /' : ; ; \ ;
1018 \      .-" .-".-" .' .'j \ / ;/
1019 \ / .-" /.' .'.' ;_:' ;
1020 :-"-. ' /-.' / '._._.'
1021 \ 't . _ /
1022 "-.t-._:'}%<<<
1023 \AddAnimal[tail-count=3]{yoda-head}%>>>
1024 { \
1025 \
1026      .-.' : '._
1027      .-'.' ; .-'.'
1028      / : _ _ \ ; / _ _ \
1029 ,'_ "----.: _;"-.": : "-.": _ _ ; _ _ " ' ,
1030 :' 't"----.' <@.' ; ,@>' .-""j.' ' ;
1031 ':-..J '-.-'L_ ' L_..;'
1032 "-._ ; .-" "-. : _ _ -"
1033 L ' / _ _ _ _ _ \ ' J
1034 "-. " _ _ .-"
1035 _ _ .l"-: _JL_:-"; . _ _
1036 .-j/' ; ; "" "" / .' \ "-.
1037 .' /:' : : /.".' ; ' .
1038 .-" / ; ' . : .". " : "-.
1039 .+"-. : : ". " . " . " ; _ _ \}%<<<
1040 %^A from https://www.ascii-code.com/ascii-art/movies/star-wars.php
1041 \AddAnimal{small-yoda}%>>>
1042 { \
1043 \
1044 _ _ .-.-
1045 '-.-"7'
1046 /' .-c
1047 | /T
1048 _)_/LI}%<<<
1049 \AddAnimal{r2d2}%>>>
1050 { \
1051 \ ,-----.
1052 ,'_ _/_/_\_'
1053 /<<::8[0]:>\
1054 _|-----|_
1055 | | ==--== | |
1056 | | --==--== | |
1057 \ | ::::|()|| /
1058 | | ...|()|| |
1059 | | _ _ _ _ _ | |
1060 | | \ _ _ _ _ _ / | |
1061 / \ / \ / \ \
1062 '---' '---' '---'}%<<<
1063 \AddAnimal{vader}%>>>
1064 { \
1065 \ / .-'~~~~~'-. \
1066 / \ \ \ \ \ \ \ \ \ \
1067 | \ \ \ \ \ \ \ \ \ |
1068 | _ _ _ _ _ | _ _ _ _ _ |
1069 |/ _ _ _ _ \ / _ _ _ _ \

```



```

1070      / ( ) ( ) \
1071     / \ ----- () ----- / \
1072     / \      /||\      / \
1073     / \     /||||\     / \
1074     / \    /|||||\    / \
1075     /- \0=====0/ -\
1076     '---...--|'-.-.-' |___...--'
1077             |      ' ' |}%<<<
1078 \AddAnimal[tail-symbol=|,tail-count=1]{crusader}%>>>
1079 { |
1080 \[T]/}
1081 \csname bool_if:cT\endcsname {l_ducksay_version_one_bool}
1082   {\AnimalOptions{crusader}{tail-1=|,rel-align=c}}
1083 \csname bool_if:cT\endcsname {l_ducksay_version_two_bool}
1084   {\AnimalOptions{crusader}{tail-1=|,body-align=c}}}%<<<
1085 %^~A http://ascii.co.uk/art/knights
1086 \AddAnimal[tail-count=3]{knight}%>>>
1087 {
1088     \ ,-" "-.
1089     \ | === |
1090     ) | (
1091     ,==\' " "/'==.
1092     .'\ (':') /'.
1093     _/_|_.' : '-._|_-\_
1094     <_>'\ : / '<_>
1095     / / >=====< / /
1096     _/ .\' / ,:-.- \/=,'
1097     / _/ |__/_v^v^v\__ \
1098     \(\) |V^V^V^V^V|\_/
1099     (\ \ \ '---|---'/
1100     \ \ \ \ -._|_,-/
1101     \ \ \ \ |__|_|
1102     \ \ \ \ <_X_>
1103     \ \ \ \ \..|../
1104     \ \ \ \ \ | /
1105     \ \ \ \ /V|V\
1106     \ | / | \
1107     '---' '---'}%<<<
1108 </animals>

```

Who's gonna use it anyway?

0

o

>( ' )

) /

/(

/ '-----/

\ ~=- /

~ ^ ~ ^ ~ ^ ~ ^ ~ ^ ~ ^ ~ ^

Hosted at  
[https://github.com/Skillmon/ltx\\_ducksay](https://github.com/Skillmon/ltx_ducksay)  
it is.

--.-.-  
'-.-"7'  
/'.-c  
| /T  
\_)\_/\_LI