

```
-----  
/ This is \  
 \ ducksay! /  
-----
```

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

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

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

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

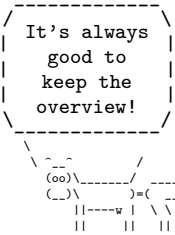
```
 \ \  
  /(\ /(\oo) /  
 /(\ /(\ )  
 | w----||  
 ||      ||
```

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

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

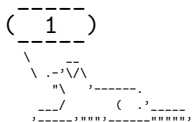
```
-----  
( Today is 2019-01-13 )  
-----
```

```
 \ .X//|X|X|  
 \ |X//|X//|X//|  
 / . 'X//|X//|X//|  
 o__ _|X//|X//|X//|
```



Contents

- 1 Documentation** **2**
- 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 `\AddAnimal` 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
-
- 2 Implementation** **15**
- 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 `\AddAnimal` 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 18
- 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

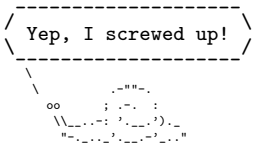


2.3.3	Options	24
2.3.4	Functions	26
2.3.4.1	Internal	26
2.3.4.1.1	Message Reading Functions	32
2.3.4.1.2	Generating Variants of External Functions	34
2.3.4.2	Document level	34
2.4	Definition of the Animals	35

1 Documentation

1.1 Downward Compatibility Issues

- v2.0 • Versions prior to v2.0 did use a regular expression for the option `ligatures`, see [subsection 1.2.2](#) for more on this issue. With v2.0 I do refer to the package's version, not the code variant which can be selected with the `version` option.
- In a document created with package versions prior to v2.0 you'll have to specify the option `version=1` in newer versions to make those old documents behave like they used to.
- v2.3 • Since v2.3 `\AddAnimal` and `\AddColoredAnimal` behave differently. You no longer have to make sure that in the first three lines every backslash which is only preceded by spaces is the bubble's tail. Instead you can specify which symbol should be the tail and how many of such symbols there are. See [subsection 1.2.1](#) for more about the current behaviour.
- The `add-think` key is deprecated and will throw an error starting with v2.3. In future versions it will be removed.



1.2 Shared between versions

1.2.1 Macros

A careful reader might notice that in the below list of macros there is no `\ducksay` and no `\duckthink` contained. This is due to differences between the two usable code variants (see the `version` key in [subsection 1.2.2](#) for the code variants, [subsection 1.3.2](#) and [subsection 1.4.2](#) for descriptions of the two macros).

`\DefaultAnimal`

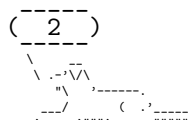
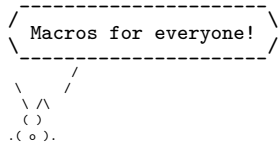
`\DefaultAnimal{<animal>}`

use the `<animal>` if none is given in the optional argument to `\ducksay` or `\duckthink`. Package default is `duck`.

`\DucksayOptions`

`\DucksayOptions{<options>}`

set the defaults to the keys described in [subsection 1.2.2](#), [subsection 1.3.3](#) and [subsection 1.4.3](#). Don't use an `<animal>` here, it has no effect.



`\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 `LATEX3`. It is therefore slower than the normal `\AddAnimal`.

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

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.

`version=number`

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

```

Options.
For every occasion
\./..
\./oo } }-e
( ) _ }
\-- } }--{ }
// _ / _ /

```

```

( 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 `o`.

`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=0`.

1.2.2.1 Options for `\AddAnimal`

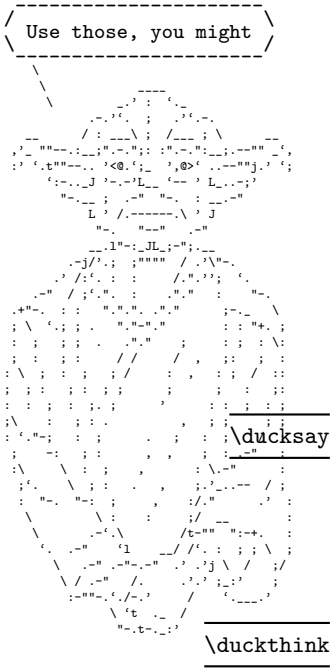
The options described here are only available in `\AddAnimal` and `\AddColoredAnimal`.

`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 Version 1

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](#).

1.3.2 Macros

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

`\ducksay` `[\options]{\message}`

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`.

`\duckthink` `[\options]{\message}`

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`.

1.3.3 Options

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

`bubble=\code`

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`

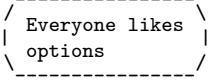
use `\valign` as the vertical alignment specifier given to the `tabular` which is around the contents of `\ducksay` and `\duckthink`.

`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`

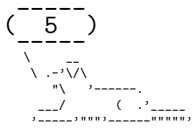
use `\column` for alignment of the bubble and the body. It should match a `tabular` column specifier. Default is `l`.



```

.N//1111.
 \N//11//1/
 /.'\N//11//1
 0...//11//11/

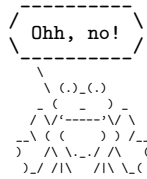
```



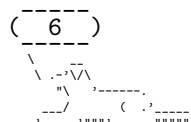
`wd=<count>` in order to detect the width the `<message>` is expanded. This might not work out for some commands (e.g. `\url` from `hyperref`). If you specify the width using `wd` the `<message>` is not expanded and therefore the command *might* work out. `<count>` should be the character count.

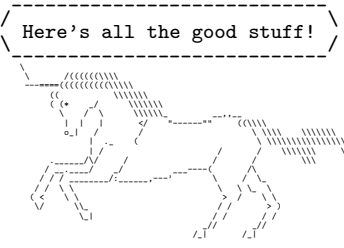
`ht=<count>` you might explicitly set the height (the row count) of the `<message>`. This only has an effect if you also specify `wd`.

1.3.4 Defects



- no automatic line wrapping





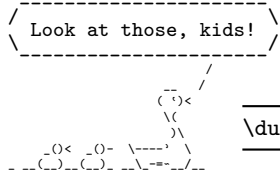
1.4 Version 2

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.

`\ducksay` `\ducksay[options]{message}`

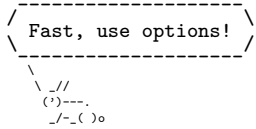
options might include any of the options described in [subsection 1.2.2](#) and [subsection 1.4.3](#) if not otherwise specified. Prints an *animal* saying *message*.

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^AT_EX₃'s coffin mechanism described in [interface3.pdf](#) and the documentation of `xcoffins`.

`\duckthink` `\duckthink[options]{message}`

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`.



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

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

`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.

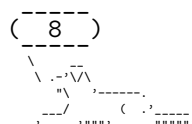
`body-x=<dimen>` defines a horizontal offset of `<dimen>` length of the `<animal>` from its placement beneath the `<message>`.

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

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

`bubble**=` clear any definitions previously made (including the package default) and set the font definitions in use to typeset the bubble to ``. 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`= $\langle token list \rangle$
the left delimiter used if only one line of delimiters is needed. Package default is (.
- `bubble-delim-left-2`= $\langle token list \rangle$
the upper most left delimiter used if more than one line of delimiters is needed. Package default is /.
- `bubble-delim-left-3`= $\langle token list \rangle$
the left delimiters used to fill the gap if more than two lines of delimiters are needed. Package default is |.
- `bubble-delim-left-4`= $\langle token list \rangle$
the lower most left delimiter used if more than one line of delimiters is needed. Package default is \.
- `bubble-delim-right-1`= $\langle token list \rangle$
the right delimiter used if only one line of delimiters is needed. Package default is).
- `bubble-delim-right-2`= $\langle token list \rangle$
the upper most right delimiter used if more than one line of delimiters is needed. Package default is \.
- `bubble-delim-right-3`= $\langle token list \rangle$
the right delimiters used to fill the gap if more than two lines of delimiters are needed. Package default is |.
- `bubble-delim-right-4`= $\langle token list \rangle$
the lower most right delimiter used if more than one line of delimiters is needed. Package default is /.
- `bubble-delim-top`= $\langle token list \rangle$
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`= $\langle dimen \rangle$
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`= $\langle dimen \rangle$
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`= $\langle column \rangle$
specifies the used column specifier used for the $\langle message \rangle$ 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`= $\langle count \rangle$
Add $\langle count \rangle$ times more `bubble-delim-top` instances than necessary to the upper and lower border of the bubble. Package default is 2.

```

-----
( 9 )
-----
 \ .-.'√\
  "      '-----
 /      ( '-----
,-----'-----

```

- `ht=<count>` specifies a minimum height (in lines) of the `<message>`. The lines' count is that of the needed lines of the horizontal bubble delimiters. If the count of the actually needed lines is smaller than the specified `<count>`, `<count>` lines will be used. Else the required lines will be used.
- `ignore-body=<bool>`
If set `true` the `<animal>`'s body will be added to the output but it will not contribute to the bounding box (so will not take up any space).
- `msg=` add `` to the font definitions in use to typeset the `<message>`.
- `msg*=` clear any definitions previously made (including the package default) and set the font definitions in use to typeset the `<message>` to ``. The package default is `\verbatim@font`.
- `MSG=` same as `msg=`, `bubble=`.
- `MSG*=` same as `msg*=`, `bubble*=`.
- `msg-align=<choice>`
specifies the alignment of the `<message>`. Possible values are `l` for flushed left, `c` for centred, `r` for flushed right and `j` for justified. If `arg=tab` or `arg=tab*` the `j` choice is only available for fixed width contents. Package default is `l`.
- `msg-align-c=<token list>`
set the `<token list>` which is responsible to typeset the message centred if the option `msg-align=c` is used. It is used independent of the `arg` key. For `arg=tab` and `arg=tab*` it is only used if a fixed width is specified and the macro `\arraybackslash` provided by `array` is used afterwards. The package default is `\centering`. It might be useful if you want to use `ragged2e`'s `\Centering` for example.
- `msg-align-j=<token list>`
set the `<token list>` which is responsible to typeset the message justified if the option `msg-align=j` is used. It is used independent of the `arg` key. For `arg=tab` and `arg=tab*` it is only used if a fixed width is specified and the macro `\arraybackslash` provided by `array` is used afterwards. The package default is empty as justification is the default behaviour of contents of a `p` column and of a `\vbox`. It might be useful if you want to use `ragged2e`'s `\justifying` for example.
- `msg-align-l=<token list>`
set the `<token list>` which is responsible to typeset the message flushed left if the option `msg-align=l` is used. It is used independent of the `arg` key. For `arg=tab` and `arg=tab*` it is only used if a fixed width is specified and the macro `\arraybackslash` provided by `array` is used afterwards. The package default is `\raggedright`. It might be useful if you want to use `ragged2e`'s `\RaggedRight` for example.
- `msg-align-r=<token list>`
set the `<token list>` which is responsible to typeset the message flushed right if the option `msg-align=r` is used. It is used independent of the `arg` key. For `arg=tab` and `arg=tab*` it is only used if a fixed width is specified and the macro `\arraybackslash` provided by `array` is used afterwards. The package default is `\raggedleft`. It might be useful if you want to use `ragged2e`'s `\RaggedLeft` for example.

```

-----
( 10 )
-----
\ .-.'√\
  "      '-----
  /      ( '-----
-----

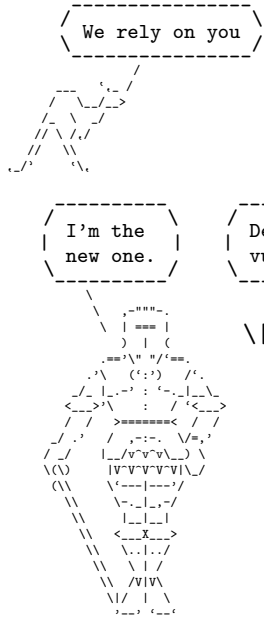
```

- `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 )
-----
 \ .-.' \ \
  "      "
-----
-----

```

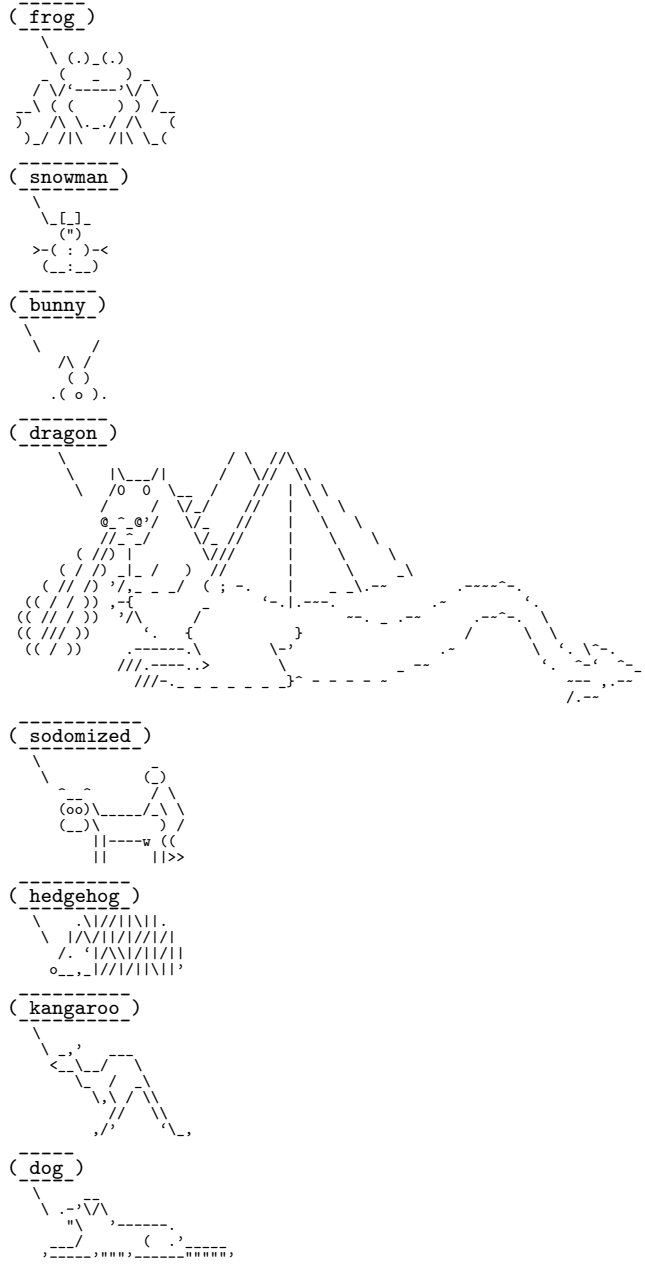
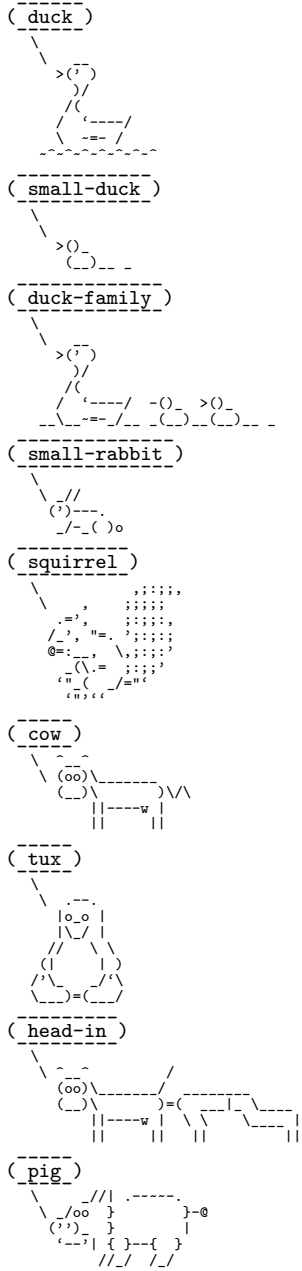


1.5 Dependencies

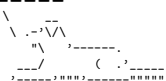
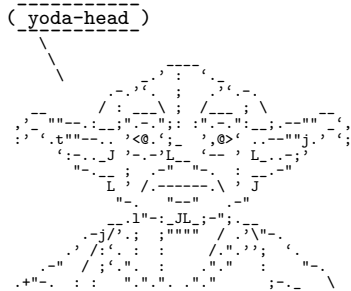
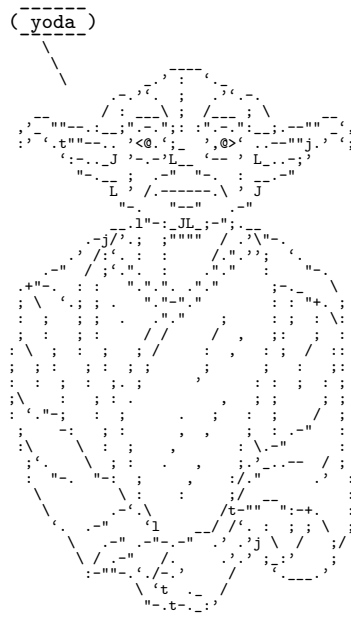
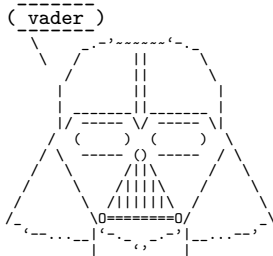
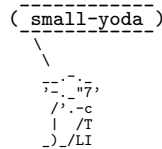
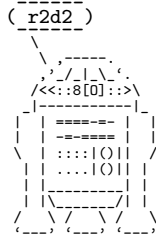
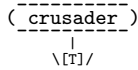
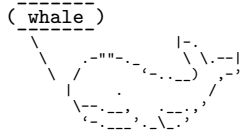
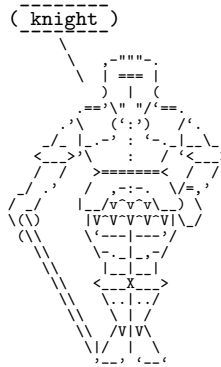
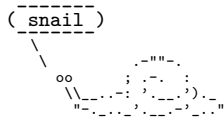
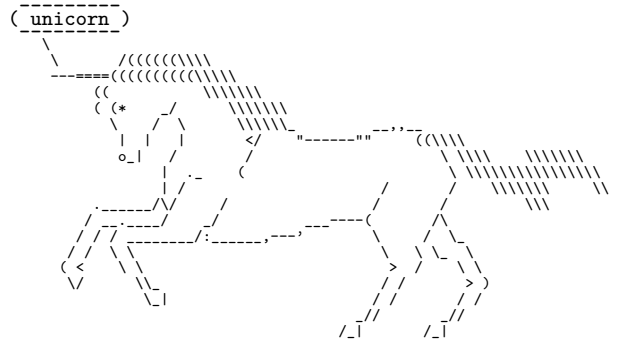
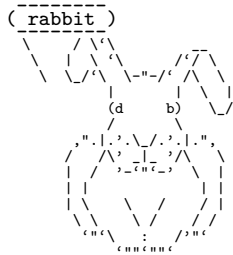
The package depends on the two packages `xparse` and `l3keys2e` and all of their dependencies. Version 2 additionally depends on `array` and `grabbox`.

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^AT_EX 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>

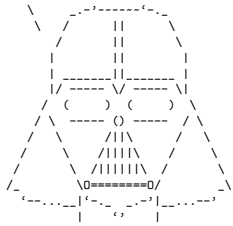
The package is hosted on https://github.com/Skillmon/ltx_ducksay, you might report bugs there.

WTFPL would be a
better license.

```

,;:::;
\ , ;:::;
 .m? , ;:::;
/_? , "m. ?;:::;
@=:__ , \; ;:::;?
_ (\ .m ;:::;?
e" ( _/m"e
e"e e e
```

Only rebel scum reads
documentation!
Join the dark side,
read the implementation.



2 Implementation

```
1 <*pkg>
```

2.1 Shared between versions

2.1.1 Variables

2.1.1.1 Integers

```
2 \int_new:N \l_ducksay_msg_width_int
3 \int_new:N \l_ducksay_msg_height_int
4 \int_new:N \l_ducksay_tail_symbol_count_int
```

2.1.1.2 Sequences

```
5 \seq_new:N \l_ducksay_msg_lines_seq
```

2.1.1.3 Token lists

```
6 \tl_new:N \l_ducksay_align_tl
7 \tl_new:N \l_ducksay_msg_align_tl
8 \tl_new:N \l_ducksay_animal_tl
9 \tl_new:N \l_ducksay_body_tl
10 \tl_new:N \l_ducksay_bubble_tl
11 \tl_new:N \l_ducksay_tmpa_tl
12 \tl_new:N \l_ducksay_tail_symbol_out_one_tl
13 \tl_new:N \l_ducksay_tail_symbol_out_two_tl
14 \tl_new:N \l_ducksay_tail_symbol_in_tl
```

2.1.1.4 Boolean

```
15 \bool_new:N \l_ducksay_version_one_bool
16 \bool_new:N \l_ducksay_version_two_bool
```

2.1.1.5 Boxes

```
17 \box_new:N \l_ducksay_tmpa_box
```

2.1.2 Regular Expressions

Regular expressions for `\AddColoredAnimal`

```
18 \regex_const:Nn \c_ducksay_textcolor_regex
19 { \c0(?:\\textcolor\{(.*)\}\{(.*)\}) }
20 \regex_const:Nn \c_ducksay_color_delim_regex
21 { \c0(?:\\bgroup\\color\{(.*)\}(.*))\\egroup }
22 \regex_const:Nn \c_ducksay_color_regex
23 { \c0(?:\\color\{(.*)\}) }
```

2.1.3 Messages

```
24 \msg_new:nnn { ducksay } { load-time-only }
25 { The~'#1'~key-is-to-be-used-only~during~package~load~time. }
26 \msg_new:nnn { ducksay } { deprecated-key }
27 { The~'\l_keys_key_tl'~key-is-deprecated.~Sorry~for~the~inconvenience. }
```

2.1.4 Key-value setup

```
28 \keys_define:nn { ducksay }
29 {
30   ,bubble .tl_set:N      = \l_ducksay_bubble_tl
31   ,body   .tl_set:N      = \l_ducksay_body_tl
```

```
( 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 \keys_define:nn { ducksay }
78 {
79 version .code:n = \msg_error:nnn { ducksay } { load-time-only } { version }
80 }

```

2.1.4.1 Keys for \AddAnimal

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

```

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

```

```

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_replace_once:Nnn { NVn }
90 \cs_generate_variant:Nn \tl_replace_all:Nnn { NVn }

```

2.1.5.2 Internal

`\ducksay_replace_verb_newline:Nn`

```

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

```

(End definition for \ducksay_replace_verb_newline:Nn. This function is documented on page ??.)

`\ducksay_replace_verb_newline_newline:Nn`

```

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

```

(End definition for \ducksay_replace_verb_newline_newline:Nn. This function is documented on page ??.)

`\ducksay_process_verb_newline:nnn`

```

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

```

(End definition for \ducksay_process_verb_newline:nnn. This function is documented on page ??.)

`\ducksay_add_animal_inner:nnnn`

```

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

```

```

( 17 )
\ .-'\
"
)
,

```

```

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

```

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

```

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

`\DucksayOptions`

```

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

```

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

`\AddAnimal`

```

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

```

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

`\AddColoredAnimal`

```

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

```

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

`\AnimalOptions`

```

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

```

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

2.1.6 Load the Correct Version and the Animals

```
200 \bool_if:NT \l_ducksay_version_one_bool
201   { \file_input:n { ducksay.code.v1.tex } }
202 \bool_if:NT \l_ducksay_version_two_bool
203   { \file_input:n { ducksay.code.v2.tex } }
204 \ExplSyntaxOff
205 \input{ducksay.animals.tex}
206 </pkg>
```

2.2 Version 1

207 `<*code.v1>`

2.2.1 Functions

2.2.1.1 Internal

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

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

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

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

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

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

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

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

```
( 21 )
\ .-'\
"
\
,-----,
|-----|
```

```

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

```

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

`\ducksay_print_msg:nn` Print out the message

```

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

```

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

```

```

294     \begin{tabular}{@{}l@{}}
295     \l_ducksay_animal_tl
296     \end{tabular}
297 \end{tabular}
298 \group_end:
299 }
300 \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

```

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

```

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

2.2.1.2 Document level

`\ducksay`

```

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

```

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

`\duckthink`

```

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

```

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

```

320 </code.v1>

```


2.3 Version 2

321 `<*code.v2>`

Load the additional dependencies of version 2.

322 `\RequirePackage{array,grabbox}`

2.3.1 Messages

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

324 `{`

325 `Justified-content-is-not-available-for-tabular-argument-mode-without-fixed-`

326 `width.~'l'~column-is-used-instead.`

327 `}`

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

329 `{`

330 `The-specified-message-alignment~'\exp_not:n { #1 }~is-unknown.~`

331 `'l'~is-used-as-fallback.`

332 `}`

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

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

2.3.2 Variables

2.3.2.1 Token Lists

335 `\tl_new:N \l_ducksay_msg_align_vbox_tl`

2.3.2.2 Boxes

336 `\box_new:N \l_ducksay_msg_box`

2.3.2.3 Bools

337 `\bool_new:N \l_ducksay_eat_arg_box_bool`

338 `\bool_new:N \l_ducksay_eat_arg_tab_verb_bool`

339 `\bool_new:N \l_ducksay_mirrored_body_bool`

2.3.2.4 Coffins

340 `\coffin_new:N \l_ducksay_body_coffin`

341 `\coffin_new:N \l_ducksay_bubble_close_coffin`

342 `\coffin_new:N \l_ducksay_bubble_open_coffin`

343 `\coffin_new:N \l_ducksay_bubble_top_coffin`

344 `\coffin_new:N \l_ducksay_msg_coffin`

2.3.2.5 Dimensions

345 `\dim_new:N \l_ducksay_hpad_dim`

346 `\dim_new:N \l_ducksay_bubble_bottom_kern_dim`

347 `\dim_new:N \l_ducksay_bubble_top_kern_dim`

348 `\dim_new:N \l_ducksay_msg_width_dim`

2.3.3 Options

349 `\keys_define:nn { ducksay }`

350 `{`

351 `,arg .choice:`

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

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

354 `{`

355 `\bool_set_false:N \l_ducksay_eat_arg_box_bool`

356 `\bool_set_false:N \l_ducksay_eat_arg_tab_verb_bool`

```

357     }
358 ,arg / tab* .code:n =
359     {
360     \bool_set_false:N \l_ducksay_eat_arg_box_bool
361     \bool_set_true:N \l_ducksay_eat_arg_tab_verb_bool
362     }
363 ,arg .initial:n = tab
364 ,wd* .dim_set:N = \l_ducksay_msg_width_dim
365 ,wd* .initial:n = -\c_max_dim
366 ,wd* .value_required:n = true
367 ,none .bool_set:N = \l_ducksay_no_body_bool
368 ,no-bubble .bool_set:N = \l_ducksay_no_bubble_bool
369 ,body-mirrored .bool_set:N = \l_ducksay_mirrored_body_bool
370 ,ignore-body .bool_set:N = \l_ducksay_ignored_body_bool
371 ,body-x .dim_set:N = \l_ducksay_body_x_offset_dim
372 ,body-x .value_required:n = true
373 ,body-y .dim_set:N = \l_ducksay_body_y_offset_dim
374 ,body-y .value_required:n = true
375 ,body-to-msg .tl_set:N = \l_ducksay_body_to_msg_align_body_tl
376 ,msg-to-body .tl_set:N = \l_ducksay_body_to_msg_align_msg_tl
377 ,body-align .choice:
378 ,body-align / l .meta:n = { body-to-msg = l , msg-to-body = l }
379 ,body-align / c .meta:n = { body-to-msg = hc , msg-to-body = hc }
380 ,body-align / r .meta:n = { body-to-msg = r , msg-to-body = r }
381 ,body-align .initial:n = l
382 ,msg-align .choice:
383 ,msg-align / l .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { l } }
384 ,msg-align / c .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { c } }
385 ,msg-align / r .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { r } }
386 ,msg-align / j .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { j } }
387 ,msg-align-l .tl_set:N = \l_ducksay_msg_align_l_tl
388 ,msg-align-l .initial:n = \raggedright
389 ,msg-align-c .tl_set:N = \l_ducksay_msg_align_c_tl
390 ,msg-align-c .initial:n = \centering
391 ,msg-align-r .tl_set:N = \l_ducksay_msg_align_r_tl
392 ,msg-align-r .initial:n = \raggedleft
393 ,msg-align-j .tl_set:N = \l_ducksay_msg_align_j_tl
394 ,msg-align-j .initial:n = {}
395 ,out-h .tl_set:N = \l_ducksay_output_h_pole_tl
396 ,out-h .initial:n = l
397 ,out-v .tl_set:N = \l_ducksay_output_v_pole_tl
398 ,out-v .initial:n = vc
399 ,out-x .dim_set:N = \l_ducksay_output_x_offset_dim
400 ,out-x .value_required:n = true
401 ,out-y .dim_set:N = \l_ducksay_output_y_offset_dim
402 ,out-y .value_required:n = true
403 ,t .meta:n = { out-v = t }
404 ,c .meta:n = { out-v = vc }
405 ,b .meta:n = { out-v = b }
406 ,body* .tl_set:N = \l_ducksay_body_fount_tl
407 ,msg* .tl_set:N = \l_ducksay_msg_fount_tl
408 ,bubble* .tl_set:N = \l_ducksay_bubble_fount_tl
409 ,body* .initial:n = \verbatim@font
410 ,msg* .initial:n = \verbatim@font

```

```

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

```

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

```

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

```

2.3.4 Functions

2.3.4.1 Internal

evaluate_message_alignment_fixed_width_common:

```

458 \cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_common:

```

```

(-----)
( 26 )
\ .-'\V\
" )
-----
-----)

```

```

459 {
460   \str_case:Vn \l_ducksay_msg_align_tl
461   {
462     { l } { \exp_not:N \l_ducksay_msg_align_l_tl }
463     { c } { \exp_not:N \l_ducksay_msg_align_c_tl }
464     { r } { \exp_not:N \l_ducksay_msg_align_r_tl }
465     { j } { \exp_not:N \l_ducksay_msg_align_j_tl }
466   }
467 }

```

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

`\ducksay_evaluate_message_alignment_fixed_width_tabular:`

```

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

```

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

`\ducksay_evaluate_message_alignment_fixed_width_vbox:`

```

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

```

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

`\ducksay_calculate_msg_width_from_int:`

```

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

```

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

`\ducksay_msg_tabular_begin:`

```

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

```

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

```

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

`\ducksay_digest_options:n`

```

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

```

```

539         \str_case:Vn \l_ducksay_msg_align_tl
540         {
541             { l }
542             { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { l } }
543             { c }
544             { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { c } }
545             { r }
546             { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { r } }
547             { j } {
548                 \msg_error:nn { ducksay } { justify-unavailable }
549                 \tl_set:Nn \l_ducksay_msg_tabular_column_tl { l }
550             }
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:`

```

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

```

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

`\ducksay_set_bubble_bottom_kern:`

```

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

```

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

`\ducksay_shipout:`

```

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

```

(30)

```

\ .-'\
"
/ (
,

```

```

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

```



```

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

```

(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`

```

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

```

```

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

```

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

`\ducksay_eat_argument_tabular_inner:w`

```

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

```

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

`\ducksay_eat_argument_tabular_verb:w`

```

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

```

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

`\ducksay_eat_argument_tabular_normal:w`

```

760 \NewDocumentCommand \ducksay_eat_argument_tabular_normal:w { +m }
761   { \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`

```

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

```

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

`\ducksay_eat_argument_vbox:w`

```

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

```

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

2.3.4.1.2 Generating Variants of External Functions

```

786 \cs_generate_variant:Nn \coffin_join:NnnNnnnn { NVnNVnnn }
787 \cs_generate_variant:Nn \coffin_attach:NnnNnnnn { NVnNVnnn }
788 \cs_generate_variant:Nn \coffin_typeset:Nnnnn { NVVnn }
789 \cs_generate_variant:Nn \str_case:nn { Vn }

```

2.3.4.2 Document level

`\ducksay`

```

790 \NewDocumentCommand \ducksay { 0{} }
791 {
792   \ducksay_digest_options:n { #1 }
793 }

```

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

`\duckthink`

```

794 \NewDocumentCommand \duckthink { 0{} }
795 {
796   \ducksay_digest_options:n { think, #1 }
797 }

```

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

```

798 </code.v2>

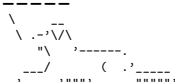
```

2.4 Definition of the Animals

```

799 <*animals>
800 %^A some of the below are from http://ascii.co.uk/art/kangaroo
801 \AddAnimal{duck}%>>>
802 { \
803   \
804     >(' )
805     )/
806     /(
807     / '----/
808     \ ~=- /
809     ~~~~~~}%<<<
810 \AddAnimal{small-duck}%>>>
811 { \
812   \
813     >()_
814     ( )__ _}%<<<
815 \AddAnimal{duck-family}%>>>
816 { \
817   \
818     >(' )
819     )/
820     /(
821     / '----/ -()_ >()_
822     __\__~=-/_ __ ( )__ ( )__ _ _}%<<<
823 \AddAnimal{cow}%>>>
824 { \
825   \ (oo)\_____
826     ( )\          )\ \
827     ||----w |
828     ||      ||}%<<<
829 \AddAnimal{head-in}%>>>
830 { \
831   \ ^--^ /
832     (oo)\_____ / _____
833     ( )\          )=( _____ | _ \_____
834     ||----w | \ \ \ \ \_____ |
835     ||      ||      ||      ||}%<<<
836 \AddAnimal{sodomized}%>>>
837 { \
838   \          ( )
839     ^--^ / \
840     (oo)\_____ / \ \
841     ( )\          ) /
842     ||----w ((
843     ||      ||>>>}%<<<
844 \AddAnimal{tux}%>>>
845 { \
846   \ .--.
847     |o_o |
848     |\_/ |
849     // \ \
850     (|   | )

```



```

851 /'\_ _/'\
852 \__)=(___/)%<<<
853 \AddAnimal{pig}%>>>
854 + \_ _//| .-----.
855 \_ /oo } }-@
856 ('')_ } |
857 '---' | { }--{ }
858 //_/ /_/+%<<<
859 \AddAnimal{frog}%>>>
860 { \
861 \ (. )_ (.)
862 _ ( _ ) _
863 / \/'-----'\ \
864 __\ ( ( ) ) /__
865 ) \ \_ . / \ (
866 )_ / | \ / | \ \_ ( ) %<<<
867 \AddAnimal{snowman}%>>>
868 { \
869 \_ [ ]_
870 (")
871 >-( : )-<
872 ( _ : _ ) %<<<
873 \AddAnimal[tail-symbol=s]{hedgehog}%>>>
874 { s .\|//| | | | |
875 s |/\| | | | | | | |
876 / . ' |/\| | | | | |
877 o _ , _ |//| | | | | ' } %<<<
878 \AddAnimal{kangaroo}%>>>
879 { \
880 \_ , '
881 <--\__/_ \
882 \_ / \_ \
883 \, \ / \ \
884 // \ \
885 ,/' ' \_ , } %<<<
886 %^A http://chris.com/ascii/index.php?art=animals/rabbits
887 \AddAnimal[tail-symbol=s,tail-count=3]{rabbit}%>>>
888 { s / \ ' \
889 s | \ \ ' \ / ' / \
890 s \_ / ' \ \ - - / ' \ \
891 | | | \ \ |
892 (d b) \_ /
893 / \
894 , ". | . ' . \_ / . ' . | . " ,
895 / \ \ ' _ | _ ' \ \ \
896 | / ' , - ' " ' - , \ \ |
897 | | | | |
898 | \ \ \ / / / / |
899 \ \ \ \ / / / / /
900 ' " ' \ : / ' " '
901 ' " " " " " ' } %<<<
902 \AddAnimal{bunny}%>>>
903 { \
904 \ /

```



```

905     /\ /
906     ( )
907     .( o ).}%<<<
908 \AddAnimal{small-rabbit}%>>>
909 { \
910   \ _//
911   (')---.
912   _/-( )o}%<<<
913 \AddAnimal[tail-symbol=s,tail-count=3]{dragon}%>>>
914 { s
915   s | \_ _/ | / \ // \ \
916   s /0 0 \_ _/ // | \ \
917   / / / \ \_ / // | \ \
918   @_~_@'/ \_ _// | \ \
919   //^_// \_ _// | \ \
920   ( // ) | \_ _// | \ \
921   ( // / ) \_ _/ // | \ \
922   ( // / ) '/, _ _/ ( ; - .
923   (( // / ) ) ,-{
924   (( // / ) ) '\ / /
925   (( // / ) ) ' . { }
926   (( // / ) ) .-----\ \_-'
927   ///.-----..> \_-'
928   ///-.-.-.-.-}^-----
929                                     /.-~}%<<<
930 %^A http://www.ascii-art.de/ascii/def/dogs.txt
931 \AddAnimal{dog}%>>>
932 { \
933   \ .-' \ \
934   " \ '-----.
935   ___/ ( . '-----
936   ,-----, " " ,-----" " " " }%<<<
937 %^A http://ascii.co.uk/art/squirrel
938 \AddAnimal{squirrel}%>>>
939 { \
940   \ , ;;;;
941   .-' , ;;;;
942   /_ ' " = . ' ;;;;
943   @ = : _ _ , \ , ;;;; '
944   _ ( \ . = ;;;; '
945   ' " _ ( _ / = " '
946   ' " ' ' }%<<<
947 \AddAnimal{snail}%>>>
948 { \
949   \ .-"-.
950   oo ; .- . :
951   \ \ _ _ . - : ' . _ . ' ) . _
952   " - . _ . _ ' . _ . - ' _ . . " }%<<<
953 %^A http://www.ascii-art.de/ascii/uvw/unicorn.txt
954 \AddAnimal{unicorn}%>>>
955 { \
956   \ /((((((\ \ \ \
957   -----((((((((((\ \ \ \
958   (( \ \ \ \ \ \ \ \

```



```

1013 \      .-'.\      /t-" " :+ . :
1014 ' .-" '1  __/ /' : ; ; \ ;
1015 \ .-" .-"-"-' .)' .)j \ / ;/
1016 \ / .-" / . .)' ;_-' ;
1017 :-"-'. /-' / '._-'
1018 \ 't .- /
1019 "-.t-._:'}%<<<
1020 \AddAnimal[tail-count=3]{yoda-head}%>>>
1021 { \
1022 \
1023 \      .-'.\      /t-" " :+ . :
1024 ' .-" '1  __/ /' : ; ; \ ;
1025 \ .-" .-"-"-' .)' .)j \ / ;/
1026 \ / .-" / . .)' ;_-' ;
1027 :-"-'. /-' / '._-'
1028 \ 't .- /
1029 "-.t-._:'}%<<<
1030 \AddAnimal{small-yoda}%>>>
1031 { \
1032 \
1033 \      .-'.\      /t-" " :+ . :
1034 ' .-" '1  __/ /' : ; ; \ ;
1035 \ .-" .-"-"-' .)' .)j \ / ;/
1036 \ / .-" / . .)' ;_-' ;
1037 :-"-'. /-' / '._-'
1038 \ 't .- /
1039 "-.t-._:'}%<<<
1040 \AddAnimal{r2d2}%>>>
1041 { \
1042 \
1043 \      .-'.\      /t-" " :+ . :
1044 ' .-" '1  __/ /' : ; ; \ ;
1045 \ .-" .-"-"-' .)' .)j \ / ;/
1046 \ / .-" / . .)' ;_-' ;
1047 :-"-'. /-' / '._-'
1048 \ 't .- /
1049 "-.t-._:'}%<<<
1050 \AddAnimal{vader}%>>>
1051 { \
1052 \
1053 \      .-'.\      /t-" " :+ . :
1054 ' .-" '1  __/ /' : ; ; \ ;
1055 \ .-" .-"-"-' .)' .)j \ / ;/
1056 \ / .-" / . .)' ;_-' ;
1057 :-"-'. /-' / '._-'
1058 \ 't .- /
1059 "-.t-._:'}%<<<

```



```
1067 / ( ) ( ) \
1068 / \ ----- () ----- / \
1069 / \ / \| \
1070 / \ / \| \| \| \| \
1071 / \ / \| \| \| \| \| \| \
1072 /_ \_ \| 0=====0/ \_ \_
1073 '---...--|'---..-'|---...--'
1074 | ' ' |}%<<<
1075 \AddAnimal[tail-symbol=|,tail-count=1]{crusader}%>>>
1076 { |
1077 \[T]/}
1078 \csname bool_if:cT\endcsname {l_ducksay_version_one_bool}
1079 {\AnimalOptions{crusader}{tail-1=|,rel-align=c}}
1080 \csname bool_if:cT\endcsname {l_ducksay_version_two_bool}
1081 {\AnimalOptions{crusader}{tail-1=|,body-align=c}}}%<<<
1082 %^A http://ascii.co.uk/art/knights
1083 \AddAnimal[tail-count=3]{knight}%>>>
1084 { \
1085 \ ,-" "-.
1086 \ | === |
1087 ) | (
1088 ,=='\ " " /'==.
1089 .'\ (':') /'.
1090 _/ _ |_-.' : '-._|_ _ \_
1091 <_>'\ : / '<_>
1092 / / >===== < / /
1093 _/ .' / ,:-. \/=,'
1094 / _/ |_/v^v^v\_) \
1095 \(\) |V^V^V^V^V|\_/
1096 (\ \ \ '---|---'/
1097 \ \ \ \-._|_-,-/
1098 \ \ \ |__|_|_|
1099 \ \ \ <_X_>
1100 \ \ \ \..|..|
1101 \ \ \ \| /
1102 \ \ \ /V|V\
1103 \ | / | \
1104 '---' '---'}%<<<
1105 </animals>
```

Who's gonna use it anyway?

0

o

> (')

) /

/(

/ '-----/

\ ~=- /

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

Hosted at
https://github.com/Skillmon/ltx_ducksay
it is.

--'-
'-_'7'
'-c
| /T
_) /LI