```
  _________
 / This is \
 \ ducksay! /
  ---------
   \
    \   __
      >(' )
        )/
       /(
      /  `----/
      \  ~=- /
    ~^~^~^~^~^~^~^
```

```
 ____________________
( But which Version? )
 --------------------
   \
    \
      >()_
       (__)__ _
```

```
                ______
               ( v2.3 )
                ------
           ^__^  /
   _______/(oo) /
/\/(       /(__)
   | w----||
   ||     ||
```

```
    ____________________________
   (  by Jonathan P. Spratte    )
    ----------------------------
                /
      .-----. /
    .'_/_|_\_`.
   /<::[O]8::>>\
  _|-----------|_
 |  | -=-==== |  |
 |  | ====-=- |  |
 \  ||()|:::: |  /
  | ||()|.... | |
  | |_________| |
  | |\_______/| |
 /   \ /   \ /   \
 `___' `___' `___'
```

```
 _____________________
( Today is 2019-01-12 )
 ---------------------
   \    .\|//||\||.
    \  |/\/||/|//|/|
      /. `|/\\|/||/||
     o__,_|//|/||\||'
 -------------
/ It's always \
|   good to   |
|  keep the   |
\  overview!  /
 -------------
  \
   \ ^__^         /
     (oo)\_______/  _________
     (__)\       )=(  ____|_ \_____
         ||----w |  \ \     \_____ |
         ||     ||   ||           ||
```

Contents

1 Documentation

```
 ____________________
/                    \
\ Yep, I screwed up! /
 --------------------
  \
   \            .-""-.
     oo       ; .-.  :
      \\__..-: '.__.').
       "-._.._'.__.-'_.."
```

1.1 Downward Compatibility Issues

v2.0 • Versions prior to v2.0 did use a regular expression for the option `ligatures`, see subsubsection 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 subsubsection 1.2.1 for more information 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.

```
 ______________________
/                      \
\ Macros for everyone! /
 ----------------------
          /
 \       /
  \ /\
   ( )
 .( o ).
```

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 subsubsection 1.2.2 for the code variants, subsubsection 1.3.2 and subsubsection 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 subsubsection 1.2.2, subsubsection 1.3.3 and subsubsection 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 signalizing 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 info 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 `\color`s.
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`.

`\AddAnimalOptions`

`\AddAnimalOptions{⟨options⟩}`

With this macro you can set the ⟨*options*⟩ exclusive to `\AddAnimal` and `\AddColoredAnimal` outside of those macros. For the available options take a look at paragraph 1.2.2.1.

`\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.

```
 _____________________
/                     \
| Options.            |
| For every occasion  |
\_____________________/
  \      _//| .-~~~~-.
   \ _/oo  }        }-@
    ('')_  }        |
     `--'| { }--{  }
          //_/  /_/
```

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. 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).

`add-think=`⟨*bool*⟩
: deprecated; will throw an error

`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`.

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

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

`think`
: Sets `tail-1=O` and `tail-2=o`.

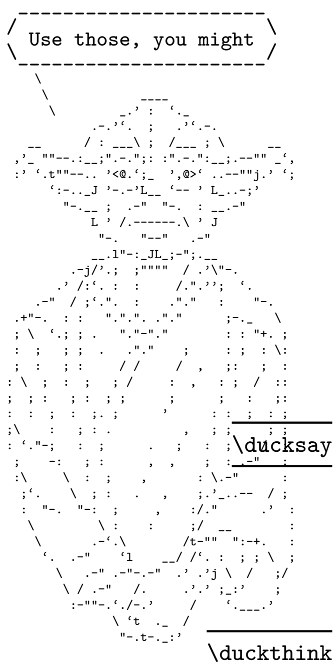
1.2.2.1 Options for `\AddAnimal`

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

`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.



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`

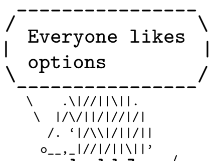
`\ducksay[⟨options⟩]{⟨message⟩}`

options might include any of the options described in subsubsection 1.2.2 and subsubsection 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`

`\duckthink[⟨options⟩]{⟨message⟩}`

options might include any of the options described in subsubsection 1.2.2 and subsubsection 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:

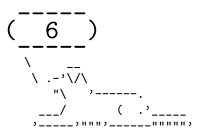
`bubble=⟨code⟩`
: use ⟨*code*⟩ in a group right before the bubble (for font switches). Might be used as a package option but not all control sequences work out of the box there.

`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`.



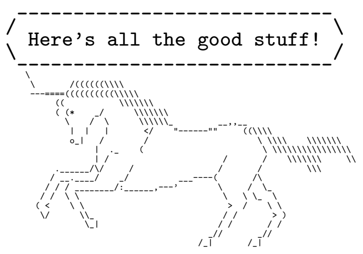
`wd=`⟨*count*⟩ in order to detect the width the ⟨*message*⟩ is expanded. This might not work out for some commands (e.g. `\url` from `hyperref`). If you specify the width using `wd` the ⟨*message*⟩ is not expanded and therefore the command *might* work out. ⟨*count*⟩ should be the character count.

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

```
 __________
/          \
\ Ohh, no! /
 ----------
   \
    \ (.)_(.)
   _ (   _   ) _
  / \/`-----'\/ \
__\ ( (     ) ) /__
)   /\ \._./ /\   (
 )_/ /|\   /|\ \_(
```

1.3.4 Defects

- no automatic line wrapping

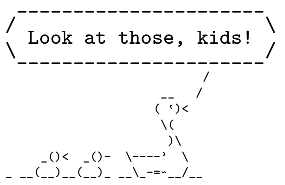


1.4 Version 2

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`, the `ligatures` and `add-think` 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.



1.4.2 Macros

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

`\ducksay`

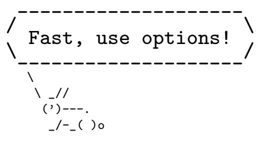
`\ducksay[⟨options⟩]{⟨message⟩}`

options might include any of the options described in subsubsection 1.2.2 and subsubsection 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 subsubsection 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 LaTeX3'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

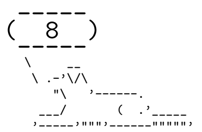
In version 2 the following options are available. Keep in mind that you shouldn't use them during package load time but in the arguments of `\ducksay`, `\duckthink` or `\DucksayOptions`.

`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=`⟨*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 necassary to the upper and lower border of the bubble. Package default is 2.

`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=`⟨*`font`*⟩ add ⟨*`font`*⟩ to the font definitions in use to typeset the ⟨*`message`*⟩.

`msg*=`⟨*`font`*⟩ clear any definitions previously made (including the package default) and set the font definitions in use to typeset the ⟨*`message`*⟩ to ⟨*`font`*⟩. The package default is `\verbatim@font`.

`MSG=`⟨*`font`*⟩ same as `msg=`⟨*`font`*⟩`, bubble=`⟨*`font`*⟩.

`MSG*=`⟨*`font`*⟩ same as `msg*=`⟨*`font`*⟩`, bubble*=`⟨*`font`*⟩.

`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.

msg-to-bubble=⟨*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.

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.

```
 ________________
/                \
\ We rely on you /
 ----------------
               /
   ___   `,_ /
  /   \__/__>
 /_  \  _/
// \ /,/
//   \\
,_/'   `\,
```

```
 __________      _______
/          \    /       \
| I'm the  |    | Deus  |
| new one. |    | vult!*|
\          /    \       /
 ----------      -------
      \             |
       \   ,-"""-.  \[T]/
        \  | === |
           )  |  (
        .=='\" "/`==.
      .'\   (`:')   /`.
    _/_ |_.-' : `-._|__\_
   <___>'\    :   / `<___>
   /  /   >=======<  /  /
 _/ .'   /  ,-:-.  \/=,'
/ _/    |__/v^v^v\__) \
\(\)     |V^V^V^V^V|\_/
 (\\     \`---|---'/
   \\     \-._|_,-/
    \\     |__|__|
     \\   <___X___>
      \\   \..|../
       \\   \ | /
        \\  /V|V\
         \|/  |  \
          '--' `--`
```

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.

```
 ______
( duck )
 ------
  \
   \   __
     >(' )
       )/
      /(
     /  `----/
     \  ~=- /
   ~^~^~^~^~^~^~^
```

```
 ____________
( small-duck )
 ------------
  \
   \
     >()_
      (__)__ _
```

```
 _____________
( duck-family )
 -------------
  \
   \   __
     >(' )
       )/
      /(
     /  `----/  -()_  >()_
   __\__~=-_/__ _(__)__(__)__ _
```

```
 ______________
( small-rabbit )
 --------------
  \
   \ _//
    (')---.
     _/-_( )o
```

```
 __________
( squirrel )
 ----------
  \               ,;:;;,
   \       ,     ;;;;;
     .=',       ;:;;:,
    /_', "=. ';:;:;
    @=:__,  \,;:;:'
      _(\.=  ;:;;'
     `"_(  _/="`
      `"'``
```

```
 _____
( cow )
 -----
  \  ^__^
   \ (oo)\_______
     (__)\       )\/\
         ||----w |
         ||     ||
```

```
 _____
( tux )
 -----
  \
   \   .--.
     |o_o |
     |\_/ |
    //   \ \
   (|     | )
  /'\_   _/`\
  \___)=(___/
```

```
 _________
( head-in )
 ---------
  \
   \ ^__^         /
     (oo)\_______/  _________
     (__)\       )=(  ____|_ \_____
         ||----w |  \ \     \_____ |
         ||     ||   ||           ||
```

```
 _____
( pig )
 -----
  \       _//| .-~~~-.
   \ _/oo  }       }-@
    ('')_  }       |
     `--'| { }--{  }
          //_/  /_/
```

```
 ______
( frog )
 ------
    \
     \ (.)_(.)
    _ (   _   ) _
   / \/`-----'\/ \
 __\ ( (     ) ) /__
 )   /\ \._./ /\   (
  )_/ /|\   /|\ \_(
```

```
 _________
( snowman )
 ---------
  \
   \_[_]_
     (")
   >-( : )-<
    (__:__)
```

```
 _______
( bunny )
 -------
  \
   \     /
     /\ /
      ( )
    .( o ).
```

```
 ________
( dragon )
 --------
      \                    / \  //\
       \    |\___/|      /   \//  \\
        \   /0  0  \__  /    //  | \ \
           /     /  \/_/    //   |  \  \
           @_^_@'/   \/_   //    |   \   \
           //_^_/     \/_ //     |    \    \
        ( //) |        \///      |     \     \
      ( / /) _|_ /   )  //       |      \     _\
    ( // /) '/,_ _ _/  ( ; -.    |    _ _\.-~        .-~~~^-.
  (( / / )) ,-{        _      `-.|.-~-.           .~         `.
 (( // / ))  '/\      /                 ~-. _ .-~      .-~^-.  \
 (( /// ))      `.   {            }                   /      \  \
  (( / ))     .----~-.\        \-'                 .~         \  `. \^-.
             ///.----..>        \             _ -~             `.  ^-`  ^-_
               ///-._ _ _ _ _ _ _}^ - - - - ~                     ~-- ,.-~
                                                                  /.-~
```

```
 ___________
( sodomized )
 -----------
  \
   \          _
             (_)
     ^__^     / \
     (oo)\_____/_\ \
     (__)\       ) /
         ||----w ((
         ||     ||>>
```

```
 __________
( hedgehog )
 ----------
  \    .\|//||\||.
   \  |/\/||/|//|/|
     /. `|/\\|/||/||
    o__,_|//|/||\||'
```

```
 __________
( kangaroo )
 ----------
  \
   \ _,'
    <__\__/   ___
       \_  /  _\
         \,\ / \\
           //   \\
         ,/'     `\_,
```

```
 _____
( dog )
 -----
  \
   \ .-'\/\
      "\    '______.
    ___/       (  .'_____
   ,_____,""",______"""""",
```

*Latin; “I’m new, too.”

```
 ---------
( rabbit )
 ---------
  \     / \`\          __
   \   |  \ `\        /`/ \
    \  \_/`\  \-"-/` /\  \
            |       |  \  |
            (d     b)   \_/
            /       \
        ,".|.'.\_/.'.|.",
       /   /\' _|_ '/\   \
       |  /  '-`"`-'  \  |
       | |             | |
       | \    \   /    / |
        \ \    \ /    / /
         `"`\   :   /'"`
             `""`""`
```

```
 ---------
( unicorn )
 ---------
  \
   \       /((((((\\\\
    ---====((((((((((\\\\\
          ((          \\\\\\\
          ( (*    _/    \\\\\\\
            \    /  \     \\\\\\_           __,,__
             |  |   |      </    "------""         ((\\\\
             o_|   /        /                        \ \\\\    \\\\\\\
                  |  ._    (                          \ \\\\\\\\\\\\\\\\
                  | /                      /          /    \\\\\\\     \\
          .______/\/     /                /          /         \\\
         / __.____/    _/         ___----(          /\
        / / /  ________/:______,---'      \        / \_
       / /  \ \                            \      \ \_  \
      ( <    \ \                            >     /   \ \
       \/     \\_                          / /        > )
               \_|                        / /        / /
                                        _//        _//
                                       /_|        /_|
```

```
 -------
( snail )
 -------
  \
   \          .-""-.
    oo       ; .-.  :
     \\__..-: '.__.')._
      "-._.._'.__.-'_.."
```

```
 ---------
( knight )
 ---------
    \
     \   ,-"""-.
      \  | === |
         )  |  (
      .=='\" "/`==.
    .'\    (`:')    /`.
  _/_ |_.-' : `-._|__\_
 <___>'\    :    / `<___>
 /  /   >=======<   /  /
_/ .'   /  ,-:-.  \/=,'
/ _/    |__/v^v^v\__) \
\(\)     |V^V^V^V^V|\_/
 (\\     \`---|---'/
   \\     \-._|_,-/
    \\     |__|__|
     \\   <___X___>
      \\   \..|../
       \\   \ | /
        \\  /V|V\
         \|/  |  \
          '--' `--`
```

```
 -------
( whale )
 -------
  \                 |-.
   \    .-""-._      \ \.--|
    \  /       `-..__)  ,-'
      |     .           /
       \--.__,   .__.,'
        `-.___'._\_.'
```

```
 ----------
( crusader )
 ----------
      |
    \[T]/
```

```
 ------
( r2d2 )
 ------
   \
    \ ,-----.
    ,'_/_|_\_`.
   /<<::8[O]::>\
  _|-----------|_
 |  | ====-=- |  |
 |  | -=-==== |  |
 \  | ::::|()||  /
  | | ....|()|| |
  | |_________| |
  | |\_______/| |
 /   \ /   \ /   \
 `---' `---' `---'
```

```
 ------------
( small-yoda )
 ------------
   \
    \
     __.-._
     '-._"7'
      /'.-c
      |  /T
     _)_/LI
```

```
 ------
( yoda )
 ------
    \
     \
      \                 ____
                     _.' :  `._
                 .-.'`.  ;   .'`.-.
        __      / : ___\ ;  /___ ; \      __
      ,'_ ""--.:__;".-.";: :".-.":__;.--"" _`,
      :' `.t""--.. '<@.`;_  ',@>` ..--""j.' `;
           `:-.._J '-.-'L__ `-- ' L_..-;'
             "-.__ ;  .-"  "-.  : __.-"
                 L ' /.------.\ ' J
                  "-.   "--"   .-"
                 __.l"-:_JL_;-";.__
              .-j/'.;  ;""""  / .'\"-.
            .' /:`. :  :     /.".'';  `.
         .-"  / ;`.". :     .".":     "-.
      .+"-.  : :   ".".". .".""       ;-._   \
      ; \  `.; ; .    ".".-"."         : : "+. ;
      :  ;   ; ;  .    ."."     ;      : ;  : \:
      ;  :   ; :      / /     /  ,    ;:   ;  :
     : \  ;  :  ;   ; /      :  ,   : ;  /  ::
     ;  ; :   ; :  ; ;       ;       ;   :   ;:
     :  :  ;  :  ;. ;       '       : :  ;  : ;
     ;\    :   ; : .          ,    ; ;     ; ;
     : `."-;   :  ;      .   ;   :  ;    /   ;
      ;    -:    ; :      ;  ,    ;  : .-"   :
      :\     \   :  ;    ,       : \.-"      :
       ;`.    \  ; :   .   ,    ;.'_..--  / ;
       :  "-.  "-:  ;     ,    :/."      .'  :
        \         \ :    :    ;/  __         :
         \       .-`.\        /t-""  ":-+.   :
          `.  .-"    `l    __/ /`. :  ; ; \  ;
            \   .-" .-"-.-"  .' .'j \  /   ;/
             \ / .-"   /.     .'.' ;_:'    ;
              :-""-.`./-.'     /    `.___.'
                    \ `t  ._  /
                     "-.t-._:'
```

```
 -------
( vader )
 -------
   \    _.-'~~~~~~`-._
    \  /      ||      \
      /       ||       \
     |        ||        |
     | _______||_______ |
     |/ ----- \/ ----- \|
    /  (     )  (     )  \
   / \  ----- () -----  / \
  /   \      /||\      /   \
 /     \    /||||\    /     \
/       \  /||||||\  /       \
/_        \O========O/        _\
  `--...__|`-._  _.-'|__...--'
          |   `'    |
```

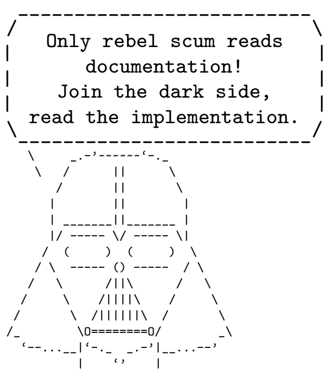
```
 -----------
( yoda-head )
 -----------
   \
    \
     \                 ____
                    _.' :  `._
                .-.'`.  ;   .'`.-.
       __      / : ___\ ;  /___ ; \      __
     ,'_ ""--.:__;".-.";: :".-.":__;.--"" _`,
     :' `.t""--.. '<@.`;_  ',@>` ..--""j.' `;
          `:-.._J '-.-'L__ `-- ' L_..-;'
            "-.__ ;  .-"  "-.  : __.-"
                L ' /.------.\ ' J
                 "-.   "--"   .-"
                __.l"-:_JL_;-";.__
             .-j/'.;  ;""""  / .'\"-.
           .' /:`. :  :     /.".'';  `.
        .-"  / ;`.". :     .".":     "-.
     .+"-.  : :   ".".". ."."         ;-._   \
```

```
 ___________________
/                   \
| WTFPL would be a  |
| better license.   |
\___________________/
  \          ,;:;;,
   \    ,    ;;;;;
     .=',    ;:;;:,
    /_', "=. ';:;:;
    @=:__,  \,;:;:'
      _(\.=  ;:;;'
     `"_(  _/="`
      `";``
```

1.7 Miscellaneous

This work may be distributed and/or modified under the conditions of the LaTeX 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.


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

2 Implementation

```
⟨*pkg⟩
```

2.1 Shared between versions

2.1.1 Variables

2.1.1.1 Integers

```
\int_new:N \l_ducksay_msg_width_int
\int_new:N \l_ducksay_msg_height_int
\int_new:N \l_ducksay_tail_symbol_count_int
```

2.1.1.2 Sequences

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

2.1.1.3 Token lists

```
\tl_new:N \l_ducksay_align_tl
\tl_new:N \l_ducksay_msg_align_tl
\tl_new:N \l_ducksay_animal_tl
\tl_new:N \l_ducksay_body_tl
\tl_new:N \l_ducksay_bubble_tl
\tl_new:N \l_ducksay_tmpa_tl
\tl_new:N \l_ducksay_tail_symbol_out_one_tl
\tl_new:N \l_ducksay_tail_symbol_out_two_tl
\tl_new:N \l_ducksay_tail_symbol_in_tl
```

2.1.1.4 Boolean

```
\bool_new:N \l_ducksay_version_one_bool
\bool_new:N \l_ducksay_version_two_bool
```

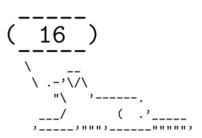
2.1.1.5 Boxes

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

2.1.2 Regular Expressions

Regular expressions for `\AddColoredAnimal`

```
\regex_const:Nn \c_ducksay_textcolor_regex
  { \cO(?:\\textcolor\{(.*?)\}\{(.*?)\}) }
\regex_const:Nn \c_ducksay_color_delim_regex
  { \cO(?:\\bgroup\\color\{(.*?)\}(.*)\\egroup) }
\regex_const:Nn \c_ducksay_color_regex
  { \cO(?:\\color\{(.*?)\}) }
```



2.1.3 Messages

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

2.1.4 Key-value setup

```
\keys_define:nn { ducksay }
  {
    ,bubble .tl_set:N      = \l_ducksay_bubble_tl
    ,body   .tl_set:N      = \l_ducksay_body_tl
    ,align  .tl_set:N      = \l_ducksay_align_tl
    ,align  .value_required:n = true
    ,wd     .int_set:N     = \l_ducksay_msg_width_int
    ,wd     .initial:n     = -\c_max_int
    ,wd     .value_required:n = true
    ,ht     .int_set:N     = \l_ducksay_msg_height_int
    ,ht     .initial:n     = -\c_max_int
    ,ht     .value_required:n = true
    ,animal .code:n        =
      { \keys_define:nn { ducksay } { default_animal .meta:n = { #1 } } }
    ,animal .initial:n     = duck
    ,msg-align .tl_set:N   = \l_ducksay_msg_align_tl
    ,msg-align .initial:n  = l
    ,msg-align .value_required:n = true
    ,rel-align .tl_set:N   = \l_ducksay_rel_align_tl
    ,rel-align .initial:n  = l
    ,rel-align .value_required:n = true
    ,ligatures .tl_set:N   = \l_ducksay_ligatures_tl
    ,ligatures .initial:n  = { `<>,'- }
    ,add-think .code:n     = \msg_error:nn { ducksay } { deprecated-key }
    ,tail-1    .tl_set:N   = \l_ducksay_tail_symbol_out_one_tl
    ,tail-1    .initial:x  = \c_backslash_str
    ,tail-2    .tl_set:N   = \l_ducksay_tail_symbol_out_two_tl
    ,tail-2    .initial:x  = \c_backslash_str
    ,no-tail   .meta:n     = { tail-1 = { ~ }, tail-2 = { ~ } }
    ,think     .meta:n     = { tail-1 = { O }, tail-2 = { o } }
    ,say       .code:n     =
      {
        \exp_args:Nx \DucksayOptions
          { tail-1 = { \c_backslash_str }, tail-2 = { \c_backslash_str } }
      }
    ,version   .choice:
    ,version / 1 .code:n   =
      {
        \bool_set_false:N \l_ducksay_version_two_bool
        \bool_set_true:N  \l_ducksay_version_one_bool
      }
    ,version / 2 .code:n   =
      {
        \bool_set_false:N \l_ducksay_version_one_bool
        \bool_set_true:N  \l_ducksay_version_two_bool
      }
    ,version    .initial:n  = 2
  }
\ProcessKeysOptions { ducksay }
```

Undefine the load-time-only keys

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

2.1.4.1 Keys for `\AddAnimal`

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

```
\keys_define:nn { ducksay / add-animal }
  {
    ,tail-symbol .code:n    =
      \tl_set:Nx \l_ducksay_tail_symbol_in_tl { \tl_to_str:n { #1 } }
    ,tail-symbol .initial:o = \c_backslash_str
    ,tail-count  .int_set:N = \l_ducksay_tail_symbol_count_int
    ,tail-count  .initial:n = 2
  }
```

2.1.5 Functions

2.1.5.1 Generating Variants of External Functions

```
\cs_generate_variant:Nn \tl_if_eq:nnT { VnT }
\cs_generate_variant:Nn \tl_replace_once:Nnn { NVn }
```

2.1.5.2 Internal

\ducksay_replace_verb_newline:Nn

```
\cs_new_protected:Npx \ducksay_replace_verb_newline:Nn #1 #2
  {
    \tl_replace_all:Nnn #1 { \char_generate:nn { 13 } { 12 } } { #2 }
  }
```

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

\ducksay_replace_verb_newline_newline:Nn

```
\cs_new_protected:Npx \ducksay_replace_verb_newline_newline:Nn #1 #2
  {
    \tl_replace_all:Nnn #1
      { \char_generate:nn { 13 } { 12 } \char_generate:nn { 13 } { 12 } } { #2 }
  }
```

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

\ducksay_process_verb_newline:nnn

```
\cs_new_protected:Npn \ducksay_process_verb_newline:nnn #1 #2 #3
  {
    \tl_set:Nn \ProcessedArgument { #3 }
    \ducksay_replace_verb_newline_newline:Nn \ProcessedArgument { #2 }
    \ducksay_replace_verb_newline:Nn \ProcessedArgument { #1 }
  }
```

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

\ducksay_add_animal_inner:nnnn

```
\cs_new_protected:Npn \ducksay_add_animal_inner:nnnn #1 #2 #3 #4
  {
    \group_begin:
      \AddAnimalOptions { #1 }
      \tl_set:Nn \l_ducksay_tmpa_tl { \ #3 }
      \int_compare:nNnTF { \l_ducksay_tail_symbol_count_int } < { \c_zero_int }
        {
          \tl_replace_once:NVn
            \l_ducksay_tmpa_tl
            \l_ducksay_tail_symbol_in_tl
            \l_ducksay_tail_symbol_out_one_tl
          \tl_replace_all:NVn
            \l_ducksay_tmpa_tl
            \l_ducksay_tail_symbol_in_tl
            \l_ducksay_tail_symbol_out_two_tl
        }
        {
          \int_compare:nNnT { \l_ducksay_tail_symbol_count_int } >
            { \c_zero_int }
            {
              \tl_replace_once:NVn
                \l_ducksay_tmpa_tl
                \l_ducksay_tail_symbol_in_tl
                \l_ducksay_tail_symbol_out_one_tl
              \int_step_inline:nnn { 2 } { \l_ducksay_tail_symbol_count_int }
                {
                  \tl_replace_once:NVn
                    \l_ducksay_tmpa_tl
                    \l_ducksay_tail_symbol_in_tl
                    \l_ducksay_tail_symbol_out_two_tl
                }
            }
        }
      \exp_args:NNNV
    \group_end:
    \tl_set:Nn \l_ducksay_tmpa_tl \l_ducksay_tmpa_tl
    \tl_map_inline:Nn \l_ducksay_ligatures_tl
      { \tl_replace_all:Nnn \l_ducksay_tmpa_tl { ##1 } { { ##1 } } }
    \ducksay_replace_verb_newline:Nn \l_ducksay_tmpa_tl { \tabularnewline\null }
    \tl_gset_eq:cN { g_ducksay_animal_#2_tl } \l_ducksay_tmpa_tl
    \exp_args:Nnx \keys_define:nn { ducksay }
      {
        #2 .code:n =
          {
            \exp_not:n { \tl_set_eq:NN \l_ducksay_animal_tl }
            \exp_after:wN \exp_not:N \cs:w g_ducksay_animal_#2_tl \cs_end:
            \exp_not:n { \exp_args:NV \DucksayOptions }
            \exp_after:wN
              \exp_not:N \cs:w l_ducksay_animal_#2_options_tl \cs_end:
          }
      }
    \tl_if_exist:cF { l_ducksay_animal_#2_options_tl }
      { \tl_new:c { l_ducksay_animal_#2_options_tl } }
    \IfBooleanT { #4 }
      { \keys_define:nn { ducksay } { default_animal .meta:n = { #2 } } }
  }
\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

```
\NewDocumentCommand \DefaultAnimal { m }
  {
    \keys_define:nn { ducksay } { default_animal .meta:n = { #1 } }
  }
```

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

\DucksayOptions

```
\NewDocumentCommand \DucksayOptions { m }
  {
    \keys_set:nn { ducksay } { #1 }
  }
```

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

\AddAnimalOptions

```
\NewDocumentCommand \AddAnimalOptions { m }
  {
    \keys_set:nn { ducksay / add-animal } { #1 }
  }
```

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

\AddAnimal

```
\NewDocumentCommand \AddAnimal { s O{} m +v }
  {
    \ducksay_add_animal_inner:nnnn { #2 } { #3 } { #4 } { #1 }
  }
```

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

\AddColoredAnimal

```
\NewDocumentCommand \AddColoredAnimal { s O{} m +v }
  {
    \tl_set:Nn \l_ducksay_tmpa_tl { #4 }
    \regex_replace_all:NnN \c_ducksay_color_delim_regex
      { \c{bgroup}\c{color}\cB\{\1\cE\}\2\c{egroup} }
      \l_ducksay_tmpa_tl
    \regex_replace_all:NnN \c_ducksay_color_regex
      { \c{color}\cB\{\1\cE\} }
      \l_ducksay_tmpa_tl
    \regex_replace_all:NnN \c_ducksay_textcolor_regex
      { \c{textcolor}\cB\{\1\cE\}\cB\{\2\cE\} }
      \l_ducksay_tmpa_tl
    \ducksay_add_animal_inner:nnVn { #2 } { #3 } \l_ducksay_tmpa_tl { #1 }
  }
```

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

`\AnimalOptions`

```
\NewDocumentCommand \AnimalOptions { s m m }
  {
    \tl_if_exist:cTF { l_ducksay_animal_#2_options_tl }
      {
        \IfBooleanTF { #1 }
          { \tl_set:cn }
          { \tl_put_right:cn }
      }
      { \tl_set:cn }
    { l_ducksay_animal_#2_options_tl } { #3, }
  }
```

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

2.1.6 Load the Correct Version and the Animals

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

\ExplSyntaxOff
\input{ducksay.animals.tex}

⟨/pkg⟩
```

2.2 Version 1

```
⟨*code.v1⟩
```

2.2.1 Functions

2.2.1.1 Internal

`\ducksay_longest_line:n`

Calculate the length of the longest line

```
\cs_new:Npn \ducksay_longest_line:n #1
  {
    \int_incr:N \l_ducksay_msg_height_int
    \exp_args:NNx \tl_set:Nn \l_ducksay_tmpa_tl { #1 }
    \regex_replace_all:nnN { \s } { \c { space } } \l_ducksay_tmpa_tl
    \int_set:Nn \l_ducksay_msg_width_int
      {
        \int_max:nn
          { \l_ducksay_msg_width_int } { \tl_count:N \l_ducksay_tmpa_tl }
      }
  }
```

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

`\ducksay_open_bubble:`

Draw the opening bracket of the bubble

```
\cs_new:Npn \ducksay_open_bubble:
  {
    \begin{tabular}{@{}l@{}}
      \null\\
      \int_compare:nNnTF { \l_ducksay_msg_height_int } = { 1 } { ( }
        {
          /
          \int_step_inline:nnn
            { 3 } { \l_ducksay_msg_height_int } { \\\kern-0.2em| }
          \\\detokenize{\ }
        }
      \\[-1ex]\null
    \end{tabular}
    \begin{tabular}{@{}l@{}}
      _\\
      \int_step_inline:nnn { 2 } { \l_ducksay_msg_height_int } { \\ } \\[-1ex]
      \mbox { - }
    \end{tabular}
  }
```

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

`\ducksay_close_bubble:`

Draw the closing bracket of the bubble

```
\cs_new:Npn \ducksay_close_bubble:
  {
    \begin{tabular}{@{}l@{}}
      _\\
      \int_step_inline:nnn { 2 } { \l_ducksay_msg_height_int } { \\ } \\[-1ex]
      { - }
    \end{tabular}
    \begin{tabular}{@{}r@{}}
      \null\\
      \int_compare:nNnTF { \l_ducksay_msg_height_int } = { 1 }
        { ) }
        {
          \detokenize {\ }
          \int_step_inline:nnn
            { 3 } { \l_ducksay_msg_height_int } { \\|\kern-0.2em }
          \\/
        }
      \\[-1ex]\null
    \end{tabular}
  }
```

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

`\ducksay_print_msg:nn` Print out the message

```
\cs_new:Npn \ducksay_print_msg:nn #1 #2
  {
    \begin{tabular}{@{} #2 @{}}
      \int_step_inline:nn { \l_ducksay_msg_width_int } { _ } \\
      #1\\[-1ex]
      \int_step_inline:nn { \l_ducksay_msg_width_int } { { - } }
    \end{tabular}
  }
\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

```
\cs_new:Npn \ducksay_print:nn #1 #2
  {
    \int_compare:nNnTF { \l_ducksay_msg_width_int } < { 0 }
      {
        \int_zero:N \l_ducksay_msg_height_int
        \seq_set_split:Nnn \l_ducksay_msg_lines_seq { \\ } { #1 }
        \seq_map_function:NN \l_ducksay_msg_lines_seq \ducksay_longest_line:n
      }
      {
        \int_compare:nNnT { \l_ducksay_msg_height_int } < { 0 }
          {
            \regex_count:nnN { \c { \\ } } { #1 } \l_ducksay_msg_height_int
            \int_incr:N \l_ducksay_msg_height_int
          }
      }
    \group_begin:
      \frenchspacing
      \verbatim@font
      \@noligs
      \begin{tabular}[\l_ducksay_align_tl]{@{}#2@{}}
        \l_ducksay_bubble_tl
        \begin{tabular}{@{}l@{}}
          \ducksay_open_bubble:
          \ducksay_print_msg:nV { #1 } \l_ducksay_msg_align_tl
          \ducksay_close_bubble:
        \end{tabular}\\
        \l_ducksay_body_tl
        \begin{tabular}{@{}l@{}}
          \l_ducksay_animal_tl
        \end{tabular}
      \end{tabular}
    \group_end:
  }
\cs_generate_variant:Nn \ducksay_print:nn { nV }
```

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

`\ducksay_prepare_say_and_think:n`

Reset some variables

```
\cs_new:Npn \ducksay_prepare_say_and_think:n #1
  {
    \int_set:Nn \l_ducksay_msg_width_int  { -\c_max_int }
    \int_set:Nn \l_ducksay_msg_height_int { -\c_max_int }
    \keys_set:nn { ducksay } { #1 }
    \tl_if_empty:NT \l_ducksay_animal_tl
      { \keys_set:nn { ducksay } { default_animal } }
  }
```

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

2.2.1.2 Document level

`\ducksay`

```
\NewDocumentCommand \ducksay { O{} m }
  {
    \group_begin:
      \ducksay_prepare_say_and_think:n { #1 }
      \ducksay_print:nV { #2 } \l_ducksay_rel_align_tl
    \group_end:
  }
```

(*End definition for* `\ducksay`. *This function is documented on page* 8.)

`\duckthink`

```
\NewDocumentCommand \duckthink { O{} m }
  {
    \group_begin:
      \ducksay_prepare_say_and_think:n { think, #1 }
      \ducksay_print:nV { #2 } \l_ducksay_rel_align_tl
    \group_end:
  }
```

(*End definition for* `\duckthink`. *This function is documented on page* 8.)

```
⟨/code.v1⟩
```

2.3 Version 2

```
⟨*code.v2⟩
```

Load the additional dependencies of version 2.

```
\RequirePackage{array,grabbox}
```

2.3.1 Messages

```
\msg_new:nnn { ducksay } { justify~unavailable }
  {
    Justified~content~is~not~available~for~tabular~argument~mode~without~fixed~
    width.~`l`~column~is~used~instead.
  }
\msg_new:nnn { ducksay } { unknown~message~alignment }
  {
    The~specified~message~alignment~`\exp_not:n { #1 }`~is~unknown.~
    `l`~is~used~as~fallback.
  }
```

2.3.2 Variables

2.3.2.1 Token Lists

```
\tl_new:N \l_ducksay_msg_align_vbox_tl
```

2.3.2.2 Boxes

```
\box_new:N \l_ducksay_msg_box
```

2.3.2.3 Bools

```
\bool_new:N \l_ducksay_eat_arg_box_bool
\bool_new:N \l_ducksay_eat_arg_tab_verb_bool
\bool_new:N \l_ducksay_mirrored_body_bool
```

2.3.2.4 Coffins

```
\coffin_new:N \l_ducksay_body_coffin
\coffin_new:N \l_ducksay_bubble_close_coffin
\coffin_new:N \l_ducksay_bubble_open_coffin
\coffin_new:N \l_ducksay_bubble_top_coffin
\coffin_new:N \l_ducksay_msg_coffin
```

2.3.2.5 Dimensions

```
\dim_new:N \l_ducksay_hpad_dim
\dim_new:N \l_ducksay_bubble_bottom_kern_dim
\dim_new:N \l_ducksay_bubble_top_kern_dim
\dim_new:N \l_ducksay_msg_width_dim
```

2.3.3 Options

```
\keys_define:nn { ducksay }
  {
    ,arg .choice:
    ,arg / box  .code:n = \bool_set_true:N  \l_ducksay_eat_arg_box_bool
    ,arg / tab  .code:n =
      {
        \bool_set_false:N \l_ducksay_eat_arg_box_bool
        \bool_set_false:N \l_ducksay_eat_arg_tab_verb_bool
      }
    ,arg / tab* .code:n =
      {
        \bool_set_false:N \l_ducksay_eat_arg_box_bool
        \bool_set_true:N  \l_ducksay_eat_arg_tab_verb_bool
      }
    ,arg .initial:n = tab
    ,wd* .dim_set:N = \l_ducksay_msg_width_dim
    ,wd* .initial:n = -\c_max_dim
    ,wd* .value_required:n = true
    ,none          .bool_set:N = \l_ducksay_no_body_bool
    ,body-mirrored .bool_set:N = \l_ducksay_mirrored_body_bool
    ,ignore-body   .bool_set:N = \l_ducksay_ignored_body_bool
    ,body-x      .dim_set:N = \l_ducksay_body_x_offset_dim
    ,body-x      .value_required:n = true
    ,body-y      .dim_set:N = \l_ducksay_body_y_offset_dim
    ,body-y      .value_required:n = true
    ,body-to-msg .tl_set:N  = \l_ducksay_body_to_msg_align_body_tl
    ,msg-to-body .tl_set:N  = \l_ducksay_body_to_msg_align_msg_tl
    ,body-align .choice:
    ,body-align / l .meta:n = { body-to-msg = l , msg-to-body = l }
    ,body-align / c .meta:n = { body-to-msg = hc , msg-to-body = hc }
    ,body-align / r .meta:n = { body-to-msg = r , msg-to-body = r }
    ,body-align .initial:n = l
    ,msg-align   .choice:
    ,msg-align  / l .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { l } }
    ,msg-align  / c .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { c } }
    ,msg-align  / r .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { r } }
    ,msg-align  / j .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { j } }
    ,msg-align-l .tl_set:N  = \l_ducksay_msg_align_l_tl
    ,msg-align-l .initial:n = \raggedright
    ,msg-align-c .tl_set:N  = \l_ducksay_msg_align_c_tl
    ,msg-align-c .initial:n = \centering
    ,msg-align-r .tl_set:N  = \l_ducksay_msg_align_r_tl
    ,msg-align-r .initial:n = \raggedleft
    ,msg-align-j .tl_set:N  = \l_ducksay_msg_align_j_tl
    ,msg-align-j .initial:n = {}
    ,out-h   .tl_set:N  = \l_ducksay_output_h_pole_tl
    ,out-h   .initial:n = l
    ,out-v   .tl_set:N  = \l_ducksay_output_v_pole_tl
    ,out-v   .initial:n = vc
    ,out-x   .dim_set:N = \l_ducksay_output_x_offset_dim
    ,out-x   .value_required:n = true
    ,out-y   .dim_set:N = \l_ducksay_output_y_offset_dim
    ,out-y   .value_required:n = true
    ,t       .meta:n    = { out-v = t }
    ,c       .meta:n    = { out-v = vc }
    ,b       .meta:n    = { out-v = b }
    ,body*   .tl_set:N  = \l_ducksay_body_fount_tl
    ,msg*    .tl_set:N  = \l_ducksay_msg_fount_tl
    ,bubble* .tl_set:N  = \l_ducksay_bubble_fount_tl
    ,body*   .initial:n = \verbatim@font
    ,msg*    .initial:n = \verbatim@font
    ,bubble* .initial:n = \verbatim@font
    ,body    .code:n    = \tl_put_right:Nn \l_ducksay_body_fount_tl   { #1 }
    ,msg     .code:n    = \tl_put_right:Nn \l_ducksay_msg_fount_tl    { #1 }
    ,bubble  .code:n    = \tl_put_right:Nn \l_ducksay_bubble_fount_tl { #1 }
    ,MSG     .meta:n    = { msg  = #1 , bubble  = #1 }
    ,MSG*    .meta:n    = { msg* = #1 , bubble* = #1 }
    ,hpad    .int_set:N = \l_ducksay_hpad_int
    ,hpad    .initial:n = 2
    ,hpad    .value_required:n = true
    ,vpad    .int_set:N = \l_ducksay_vpad_int
    ,vpad    .value_required:n = true
    ,col     .tl_set:N  = \l_ducksay_msg_tabular_column_tl
    ,bubble-top-kern  .tl_set:N  = \l_ducksay_bubble_top_kern_tl
    ,bubble-top-kern  .initial:n = { -.5ex }
    ,bubble-top-kern  .value_required:n = true
    ,bubble-bot-kern  .tl_set:N  = \l_ducksay_bubble_bottom_kern_tl
    ,bubble-bot-kern  .initial:n = { .2ex }
    ,bubble-bot-kern  .value_required:n = true
    ,bubble-side-kern .tl_set:N  = \l_ducksay_bubble_side_kern_tl
    ,bubble-side-kern .initial:n = { .2em }
    ,bubble-side-kern .value_required:n = true
    ,bubble-delim-top     .tl_set:N  = \l_ducksay_bubble_delim_top_tl
    ,bubble-delim-left-1  .tl_set:N  = \l_ducksay_bubble_delim_left_a_tl
    ,bubble-delim-left-2  .tl_set:N  = \l_ducksay_bubble_delim_left_b_tl
    ,bubble-delim-left-3  .tl_set:N  = \l_ducksay_bubble_delim_left_c_tl
    ,bubble-delim-left-4  .tl_set:N  = \l_ducksay_bubble_delim_left_d_tl
    ,bubble-delim-right-1 .tl_set:N  = \l_ducksay_bubble_delim_right_a_tl
    ,bubble-delim-right-2 .tl_set:N  = \l_ducksay_bubble_delim_right_b_tl
    ,bubble-delim-right-3 .tl_set:N  = \l_ducksay_bubble_delim_right_c_tl
    ,bubble-delim-right-4 .tl_set:N  = \l_ducksay_bubble_delim_right_d_tl
    ,bubble-delim-top     .initial:n = { { - } }
    ,bubble-delim-left-1  .initial:n = (
    ,bubble-delim-left-2  .initial:n = /
    ,bubble-delim-left-3  .initial:n = |
    ,bubble-delim-left-4  .initial:n = \c_backslash_str
    ,bubble-delim-right-1 .initial:n = )
    ,bubble-delim-right-2 .initial:n = \c_backslash_str
    ,bubble-delim-right-3 .initial:n = |
    ,bubble-delim-right-4 .initial:n = /
    ,strip-spaces .bool_set:N  = \l_ducksay_msg_strip_spaces_bool
  }
```

2.3.4 Functions

2.3.4.1 Internal

aluate_message_alignment_fixed_width_common:

```
\cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_common:
  {
    \str_case:Vn \l_ducksay_msg_align_tl
      {
        { l } { \exp_not:N \l_ducksay_msg_align_l_tl }
        { c } { \exp_not:N \l_ducksay_msg_align_c_tl }
        { r } { \exp_not:N \l_ducksay_msg_align_r_tl }
        { j } { \exp_not:N \l_ducksay_msg_align_j_tl }
      }
  }
```

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

uate_message_alignment_fixed_width_tabular:

```
\cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_tabular:
  {
    \tl_if_empty:NT \l_ducksay_msg_tabular_column_tl
      {
        \tl_set:Nx \l_ducksay_msg_tabular_column_tl
          {
            >
            {
              \ducksay_evaluate_message_alignment_fixed_width_common:
              \exp_not:N \arraybackslash
            }
            p { \exp_not:N \l_ducksay_msg_width_dim }
          }
      }
  }
```

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

evaluate_message_alignment_fixed_width_vbox:

```
\cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_vbox:
  {
    \tl_set:Nx \l_ducksay_msg_align_vbox_tl
      { \ducksay_evaluate_message_alignment_fixed_width_common: }
  }
```

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

\ducksay_calculate_msg_width_from_int:

```
\cs_new:Npn \ducksay_calculate_msg_width_from_int:
  {
    \hbox_set:Nn \l_ducksay_tmpa_box { \l_ducksay_msg_fount_tl M }
    \dim_set:Nn \l_ducksay_msg_width_dim
      { \l_ducksay_msg_width_int \box_wd:N \l_ducksay_tmpa_box }
  }
```

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

`\ducksay_msg_tabular_begin:`

```
\cs_new:Npn \ducksay_msg_tabular_begin:
  {
    \ducksay_msg_tabular_begin_inner:V \l_ducksay_msg_tabular_column_tl
  }
\cs_new:Npn \ducksay_msg_tabular_begin_inner:n #1
  {
    \begin { tabular } { @{} #1 @{} }
  }
\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:`

```
\cs_new:Npn \ducksay_msg_tabular_end:
  {
    \end { tabular }
  }
```

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

`\ducksay_digest_options:n`

```
\cs_new:Npn \ducksay_digest_options:n #1
  {
    \keys_set:nn { ducksay } { #1 }
    \tl_if_empty:NT \l_ducksay_animal_tl
      { \keys_set:nn { ducksay } { default_animal } }
    \bool_if:NTF \l_ducksay_eat_arg_box_bool
      {
        \dim_compare:nNnTF { \l_ducksay_msg_width_dim } < { \c_zero_dim }
          {
            \int_compare:nNnTF { \l_ducksay_msg_width_int } < { \c_zero_int }
              {
                \cs_set_eq:NN
                  \ducksay_eat_argument:w \ducksay_eat_argument_hbox:w
              }
              {
                \cs_set_eq:NN
                  \ducksay_eat_argument:w \ducksay_eat_argument_vbox:w
                \ducksay_calculate_msg_width_from_int:
              }
          }
          {
            \cs_set_eq:NN \ducksay_eat_argument:w \ducksay_eat_argument_vbox:w
          }
      }
      {
        \dim_compare:nNnTF { \l_ducksay_msg_width_dim } < { \c_zero_dim }
          {
            \int_compare:nNnTF { \l_ducksay_msg_width_int } < { \c_zero_int }
              {
                \tl_if_empty:NT \l_ducksay_msg_tabular_column_tl
                  {
                    \str_case:Vn \l_ducksay_msg_align_tl
```

```
                      {
                        { l }
                          { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { l } }
                        { c }
                          { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { c } }
                        { r }
                          { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { r } }
                        { j } {
                          \msg_error:nn { ducksay } { justify~unavailable }
                          \tl_set:Nn \l_ducksay_msg_tabular_column_tl { l }
                        }
                      }
                  }
              }
              {
                \ducksay_calculate_msg_width_from_int:
                \ducksay_evaluate_message_alignment_fixed_width_tabular:
              }
          }
          {
            \ducksay_evaluate_message_alignment_fixed_width_tabular:
          }
        \cs_set_eq:NN \ducksay_eat_argument:w \ducksay_eat_argument_tabular:w
      }
  }
```

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

\ducksay_set_bubble_top_kern:

```
\cs_new:Npn \ducksay_set_bubble_top_kern:
  {
    \group_begin:
    \l_ducksay_bubble_fount_tl
    \exp_args:NNNx
    \group_end:
    \dim_set:Nn \l_ducksay_bubble_top_kern_dim
      { \dim_eval:n { \l_ducksay_bubble_top_kern_tl } }
  }
```

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

\ducksay_set_bubble_bottom_kern:

```
\cs_new:Npn \ducksay_set_bubble_bottom_kern:
  {
    \group_begin:
    \l_ducksay_bubble_fount_tl
    \exp_args:NNNx
    \group_end:
    \dim_set:Nn \l_ducksay_bubble_bottom_kern_dim
      { \dim_eval:n { \l_ducksay_bubble_bottom_kern_tl } }
  }
```

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

\ducksay_shipout:

```
\cs_new_protected:Npn \ducksay_shipout:
  {
    \hbox_set:Nn \l_ducksay_tmpa_box
      { \l_ducksay_bubble_fount_tl \l_ducksay_bubble_delim_top_tl }
    \int_set:Nn \l_ducksay_msg_width_int
      {
        \fp_eval:n
          {
            ceil
              ( \box_wd:N \l_ducksay_msg_box / \box_wd:N \l_ducksay_tmpa_box )
          }
      }
    \group_begin:
    \l_ducksay_bubble_fount_tl
    \exp_args:NNNx
    \group_end:
    \int_set:Nn \l_ducksay_msg_height_int
      {
        \int_max:nn
          {
            \fp_eval:n
              {
                ceil
                  (
                    (
                      \box_ht:N \l_ducksay_msg_box
                      + \box_dp:N \l_ducksay_msg_box
                    )
                    / ( \arraystretch * \baselineskip )
                  )
              }
            + \l_ducksay_vpad_int
          }
          { \l_ducksay_msg_height_int }
      }
    \hcoffin_set:Nn \l_ducksay_bubble_open_coffin
      {
        \l_ducksay_bubble_fount_tl
        \begin{tabular}{@{}l@{}}
          \int_compare:nNnTF { \l_ducksay_msg_height_int } = { \c_one_int }
            {
              \l_ducksay_bubble_delim_left_a_tl
            }
            {
              \l_ducksay_bubble_delim_left_b_tl\\
              \int_step_inline:nnn
                { 3 } { \l_ducksay_msg_height_int }
                {
                  \kern-\l_ducksay_bubble_side_kern_tl
                  \l_ducksay_bubble_delim_left_c_tl
                  \\
                }
              \l_ducksay_bubble_delim_left_d_tl
        }
      \end{tabular}
    }
    \hcoffin_set:Nn \l_ducksay_bubble_close_coffin
      {
        \l_ducksay_bubble_fount_tl
        \begin{tabular}{@{}r@{}}
          \int_compare:nNnTF { \l_ducksay_msg_height_int } = { \c_one_int }
            {
              \l_ducksay_bubble_delim_right_a_tl
            }
            {
              \l_ducksay_bubble_delim_right_b_tl \\
              \int_step_inline:nnn
                { 3 } { \l_ducksay_msg_height_int }
                {
                  \l_ducksay_bubble_delim_right_c_tl
                  \kern-\l_ducksay_bubble_side_kern_tl
                  \\
                }
              \l_ducksay_bubble_delim_right_d_tl
            }
        \end{tabular}
      }
    \hcoffin_set:Nn \l_ducksay_bubble_top_coffin
      {
        \l_ducksay_bubble_fount_tl
        \int_step_inline:nn { \l_ducksay_msg_width_int + \l_ducksay_hpad_int }
          { \l_ducksay_bubble_delim_top_tl }
      }
    \hcoffin_set:Nn \l_ducksay_msg_coffin { \box_use:N \l_ducksay_msg_box }
    \bool_if:NF \l_ducksay_no_body_bool
      {
        \hcoffin_set:Nn \l_ducksay_body_coffin
          {
            \frenchspacing
            \l_ducksay_body_fount_tl
            \begin{tabular} { @{} l @{} }
              \l_ducksay_animal_tl
            \end{tabular}
          }
        \bool_if:NT \l_ducksay_mirrored_body_bool
          {
            \coffin_scale:Nnn \l_ducksay_body_coffin
              { -\c_one_int } { \c_one_int }
            \str_case:Vn \l_ducksay_body_to_msg_align_body_tl
              {
                { l } { \tl_set:Nn \l_ducksay_body_to_msg_align_body_tl { r } }
                { r } { \tl_set:Nn \l_ducksay_body_to_msg_align_body_tl { l } }
              }
          }
      }
    \dim_set:Nn \l_ducksay_hpad_dim
      {
        (
          \coffin_wd:N \l_ducksay_bubble_top_coffin
          - \coffin_wd:N \l_ducksay_msg_coffin
        ) / 2
      }
    \coffin_join:NnnNnnnn
      \l_ducksay_msg_coffin         { l } { vc }
      \l_ducksay_bubble_open_coffin { r } { vc }
      { - \l_ducksay_hpad_dim } { \c_zero_dim }
    \coffin_join:NnnNnnnn
      \l_ducksay_msg_coffin          { r } { vc }
      \l_ducksay_bubble_close_coffin { l } { vc }
      { \l_ducksay_hpad_dim } { \c_zero_dim }
    \ducksay_set_bubble_top_kern:
    \ducksay_set_bubble_bottom_kern:
    \coffin_join:NnnNnnnn
      \l_ducksay_msg_coffin        { hc } { t }
      \l_ducksay_bubble_top_coffin { hc } { b }
      { \c_zero_dim } { \l_ducksay_bubble_top_kern_dim }
    \coffin_join:NnnNnnnn
      \l_ducksay_msg_coffin        { hc } { b }
      \l_ducksay_bubble_top_coffin { hc } { t }
      { \c_zero_dim } { \l_ducksay_bubble_bottom_kern_dim }
    \bool_if:NF \l_ducksay_no_body_bool
      {
        \bool_if:NTF \l_ducksay_ignored_body_bool
          { \coffin_attach:NVnNVnnn }
          { \coffin_join:NVnNVnnn   }
          \l_ducksay_msg_coffin  \l_ducksay_body_to_msg_align_msg_tl  { b }
          \l_ducksay_body_coffin \l_ducksay_body_to_msg_align_body_tl { t }
          { \l_ducksay_body_x_offset_dim } { \l_ducksay_body_y_offset_dim }
      }
    \coffin_typeset:NVVnn \l_ducksay_msg_coffin
      \l_ducksay_output_h_pole_tl \l_ducksay_output_v_pole_tl
      { \l_ducksay_output_x_offset_dim } { \l_ducksay_output_y_offset_dim }
    \group_end:
  }
```

(*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`

```
\cs_new:Npn \ducksay_eat_argument_tabular:w
  {
    \bool_if:NTF \l_ducksay_eat_arg_tab_verb_bool
      { \ducksay_eat_argument_tabular_verb:w }
      { \ducksay_eat_argument_tabular_normal:w }
  }
```

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

\ducksay_eat_argument_tabular_inner:w

```
\cs_new:Npn \ducksay_eat_argument_tabular_inner:w #1
  {
    \hbox_set:Nn \l_ducksay_msg_box
      {
        \l_ducksay_msg_fount_tl
        \ducksay_msg_tabular_begin:
          #1
        \ducksay_msg_tabular_end:
      }
    \ducksay_shipout:
  }
```

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

\ducksay_eat_argument_tabular_verb:w

```
\NewDocumentCommand \ducksay_eat_argument_tabular_verb:w
  { >{ \ducksay_process_verb_newline:nnn { ~ } { ~ \par } } +v }
  {
    \ducksay_eat_argument_tabular_inner:w
      {
        \group_begin:
          \tex_everyeof:D { \exp_not:N }
          \exp_after:wN
        \group_end:
        \tex_scantokens:D { #1 }
      }
  }
```

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

\ducksay_eat_argument_tabular_normal:w

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

```
\cs_new_protected_nopar:Npn \ducksay_eat_argument_hbox:w
  {
    \bool_if:NTF \l_ducksay_msg_strip_spaces_bool
      { \grabbox }
      { \grabbox* }
      \l_ducksay_msg_box [ \l_ducksay_msg_fount_tl ] \hbox \ducksay_shipout:
  }
```

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

`\ducksay_eat_argument_vbox:w`

```
\cs_new_protected_nopar:Npn \ducksay_eat_argument_vbox:w
  {
    \ducksay_evaluate_message_alignment_fixed_width_vbox:
    \bool_if:NTF \l_ducksay_msg_strip_spaces_bool
      { \grabbox }
      { \grabbox* }
      [
        \hsize \l_ducksay_msg_width_dim
        \linewidth \hsize
        \l_ducksay_msg_fount_tl
        \l_ducksay_msg_align_vbox_tl
        \@afterindentfalse
        \@afterheading
      ]
      \l_ducksay_msg_box
      \vbox \ducksay_shipout:
  }
```

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

2.3.4.1.2 Generating Variants of External Functions

```
\cs_generate_variant:Nn \coffin_join:NnnNnnnn { NVnNVnnn }
\cs_generate_variant:Nn \coffin_attach:NnnNnnnn { NVnNVnnn }
\cs_generate_variant:Nn \coffin_typeset:Nnnnn { NVVnn }
\cs_generate_variant:Nn \tl_if_eq:nnT { VnT }
\cs_generate_variant:Nn \str_case:nn { Vn }
\cs_generate_variant:Nn \regex_replace_all:NnN { Nnc }
```

2.3.4.2 Document level

`\ducksay`

```
\NewDocumentCommand \ducksay { O{} }
  {
    \group_begin:
      \ducksay_digest_options:n { #1 }
      \ducksay_eat_argument:w
  }
```

(*End definition for* `\ducksay`. *This function is documented on page 8.*)

`\duckthink`

```
\NewDocumentCommand \duckthink { O{} }
  {
    \group_begin:
      \ducksay_digest_options:n { think, #1 }
      \ducksay_eat_argument:w
  }
```

(*End definition for* `\duckthink`. *This function is documented on page 8.*)

```
⟨/code.v2⟩
```

2.4 Definition of the Animals

```
⟨*animals⟩
%^^A some of the below are from http://ascii.co.uk/art/kangaroo
\AddAnimal{duck}%>>>
{  \
    \   __
      >(' )
        )/
       /(
      /  `----/
      \  ~=- /
    ~^~^~^~^~^~^~^}%<<<
\AddAnimal{small-duck}%>>>
{  \
    \
      >()_
       (__)__ _}%<<<
\AddAnimal{duck-family}%>>>
{  \
    \   __
      >(' )
        )/
       /(
      /  `----/  -()_  >()_
    __\__~=-_/__ _(__)__(__)__ _}%<<<
\AddAnimal{cow}%>>>
{  \    ^__^
    \ (oo)\_______
      (__)\       )\/\
          ||----w |
          ||     ||}%<<<
\AddAnimal{head-in}%>>>
{  \
    \ ^__^         /
      (oo)\_______/  ________
      (__)\       )=(  ___|_ \____
          ||----w |  \ \    \____ |
          ||     ||   ||          ||}%<<<
\AddAnimal{sodomized}%>>>
{  \             _
    \           (_)
      ^__^       / \
      (oo)\_____/_\ \
      (__)\       ) /
          ||----w ((
          ||     ||>>}%<<<
\AddAnimal{tux}%>>>
{  \
    \  .--.
      |o_o |
      |\_/ |
     //   \ \
    (|     | )
```

```
   /'\_   _/`\
   \___)=(___/}%<<<
\AddAnimal{pig}%>>>
+  \     _//| .-~~~~-.
    \ _/oo  }        }-@
     ('')_  }        |
      `--'| { }--{  }
           //_/  /_/+%<<<
\AddAnimal{frog}%>>>
{   \
     \ (.)_(.)
    _ (   _   ) _
   / \/`-----'\/ \
 __\ ( (     ) ) /__
 )   /\ \._./ /\   (
  )_/ /|\   /|\ \_(}%<<<
\AddAnimal{snowman}%>>>
{  \
    \_[_]_
      (")
   >-( : )-<
    (__:__)}%<<<
\AddAnimal[tail-symbol=s]{hedgehog}%>>>
{  s    .\|//||\||.
    s  |/\/||/|//|/|
      /. `|/\\|/||/||
     o__,_|//|/||\||'}%<<<
\AddAnimal{kangaroo}%>>>
{  \
    \ _,'   ___
     <__\__/   \
        \_  /  _\
          \,\ / \\
            //   \\
          ,/'     `\_,}%<<<
%^^A http://chris.com/ascii/index.php?art=animals/rabbits
\AddAnimal[tail-symbol=s,tail-count=3]{rabbit}%>>>
{ s     / \`\         __
   s   |  \ `\      /`/ \
    s  \_/`\  \-"-/` /\  \
            |       |  \  |
            (d     b)   \_/
            /       \
        ,".|.'.\_/.'.|.",
       /   /\' _|_ '/\   \
       |  /  '_`"`_'  \  |
       | |             | |
       | \    \   /    / |
        \ \    \ /    / /
         `"`\   :   /'"`
             `""`""`}%<<<
\AddAnimal{bunny}%>>>
{ \
   \      /
```

```
      /\ /
       ( )
     .( o ).}%<<<
\AddAnimal{small-rabbit}%>>>
{  \
    \ _//
     (')---.
      _/-_( )o}%<<<
\AddAnimal[tail-symbol=s,tail-count=3]{dragon}%>>>
{     s                   / \  //\
       s    |\___/|      /   \//  \\
        s   /0  0  \__  /    //  | \ \
           /     /  \/_/    //   |  \  \
           @_^_@'/   \/_   //    |   \   \
           //_^_/     \/_ //     |    \    \
        ( //) |        \///      |     \     \
      ( / /) _|_ /   )  //       |      \     _\
    ( // /) '/,_ _ _/  ( ; -.    |    _ _\.-~        .-~~~^-.
  (( / / )) ,-{        _      `-.|.-~-.           .~         `.
 (( // / ))  '/\      /                 ~-. _ .-~      .-~^-.  \
 (( /// ))      `.   {            }                   /      \  \
  (( / ))     .----~-.\        \-'                 .~         \  `. \^-.
             ///.----..>        \             _ -~             `.  ^-`  ^-_
               ///-._ _ _ _ _ _ _}^ - - - - ~                     ~-- ,.-~
                                                                  /.-~}%<<<
%^^A http://www.ascii-art.de/ascii/def/dogs.txt
\AddAnimal{dog}%>>>
{  \     __
    \ .-'\/\
       "\   '------.
     ___/       (  .'_____
    '-----'"""'------"""""'}%<<<
%^^A http://ascii.co.uk/art/squirrel
\AddAnimal{squirrel}%>>>
{  \           ,;:;;,
    \    ,    ;;;;;
      .=',    ;:;;:,
     /_', "=. ';:;:;
     @=:__,  \,;:;:'
       _(\.=  ;:;;'
      `"_(  _/="`
       `"'``}%<<<
\AddAnimal{snail}%>>>
{  \
    \          .-""-.
      oo      ; .-.  :
       \\__..-: '.__.')._
        "-._.._'.__.-'_.."}%<<<
%^^A http://www.ascii-art.de/ascii/uvw/unicorn.txt
\AddAnimal{unicorn}%>>>
{   \
     \       /((((((\\\\
     ---====((((((((((\\\\\
          ((           \\\\\\\
```

```
          ( (*    _/      \\\\\\\
            \    /  \      \\\\\\_         __,,__
             |  |   |       </    "------""     ((\\\\
             o_|   /        /                      \ \\\\    \\\\\\\
                  |  ._    (                        \ \\\\\\\\\\\\\\\\
                  | /                       /       /    \\\\\\\     \\
          .______/\/     /                 /       /          \\\
         / __.____/    _/          ___----(       /\
        / / / ________/:______,---'        \     /  \_
       / /  \ \                             \   \ \_  \
      ( <    \ \                             >  /    \ \
       \/      \\_                          / /       > )
                \_|                        / /       / /
                                         _//       _//
                                       /_|       /_|}%<<<
%^^A https://asciiart.website//index.php?art=animals/other%20(water)
\AddAnimal[tail-count=3,tail-symbol=s]{whale}%>>>
{  s                |-.
    s    .-""-._     \ \.--|
     s  /       `-..__)  ,-'
       |     .          /
        \--.__,   .__.,'
         `-.___'._\_.'}%<<<
%^^A from http://www.ascii-art.de/ascii/s/starwars.txt :
\AddAnimal[tail-count=3]{yoda}%>>>
{   \
     \             ____
      \         _.' :  `._
            .-.'`.  ;   .'`.-.
   __      / : ___\ ;  /___ ; \      __
 ,'_ ""--.:__;".-.";: :".-.":__;.--"" _`,
 :' `.t""--.. '<@.`;_  ',@>` ..--""j.' `;
      `:-.._J '-.-'L__ `-- ' L_..-;'
        "-.__ ;  .-"  "-.  : __.-"
            L ' /.------.\ ' J
             "-.   "--"   .-"
            __.l"-:_JL_;-";.__
         .-j/'.;  ;""""  / .'\"-.
       .' /:`. :  :     /.".'';  `.
    .-"  / ;`.".  :    ."."   :    "-.
 .+"-.  : :   ".".". ."."      ;-._   \
 ; \  `.; ; .   "."-"."         : : "+. ;
 :  ;   ; ;  .   ."."    ;      : ;  : \:
 ;  :   ; :     / /     /  ,   ;:   ;  :
: \  ;  :  ;   ; /     :  ,   : ;  /  ::
;  ; :   ; :  ; ;       ;      ;   :   ;:
:  :  ;  :  ;. ;      '      : :  ;  : ;
;\    :   ; : .          ,   ; ;     ; ;
: `."-;   :  ;      .   ;   :  ;    /  ;
 ;    -:   ; :      ,  ,    ;  : .-"   :
 :\     \  :  ;    ,       : \.-"      :
  ;`.    \  ; :   .   ,    ;.'_..--  / ;
  :  "-.  "-:  ;     ,    :/."       .'  :
   \         \ :    :     ;/  __         :
    \       .-`.\        /t-""  ":-+.   :
     `.  .-"    `l    __/ /`. :  ; ; \  ;
       \   .-" .-"-.-"  .' .'j \  /   ;/
        \ / .-"   /.     .'.' ;_:'    ;
         :-""-.`./-.'     /    `.___.'
               \ `t  ._  /
                "-.t-._:'}%<<<
\AddAnimal[tail-count=3]{yoda-head}%>>>
{   \
     \             ____
      \         _.' :  `._
            .-.'`.  ;   .'`.-.
   __      / : ___\ ;  /___ ; \      __
 ,'_ ""--.:__;".-.";: :".-.":__;.--"" _`,
 :' `.t""--.. '<@.`;_  ',@>` ..--""j.' `;
      `:-.._J '-.-'L__ `-- ' L_..-;'
        "-.__ ;  .-"  "-.  : __.-"
            L ' /.------.\ ' J
             "-.   "--"   .-"
            __.l"-:_JL_;-";.__
         .-j/'.;  ;""""  / .'\"-.
       .' /:`. :  :     /.".'';  `.
    .-"  / ;`.".  :    ."."   :    "-.
 .+"-.  : :   ".".". ."."      ;-._   \}%<<<
%^^A from https://www.ascii-code.com/ascii-art/movies/star-wars.php
\AddAnimal{small-yoda}%>>>
{  \
    \
    __.-._
    '-._"7'
     /'.-c
     |  /T
    _)_/LI}%<<<
\AddAnimal{r2d2}%>>>
{  \
    \ ,-----.
    ,'_/_|_\_`.
   /<<::8[O]::>\
  _|-----------|_
 |  | ====-=- |  |
 |  | -=-==== |  |
 \  | ::::|()||  /
  | | ....|()|| |
  | |_________| |
  | |\_______/| |
 /   \ /   \ /   \
 `___' `___' `___'}%<<<
\AddAnimal{vader}%>>>
{  \     _.-'~~~~~~`-._
    \   /      ||      \
       /       ||       \
      |        ||        |
      | _______||_______ |
      |/ ----- \/ ----- \|
```

```
     /  (     )  (     )  \
    / \  ----- () -----  / \
   /   \      /||\      /   \
  /     \    /||||\    /     \
 /       \  /||||||\  /       \
/_        \O========O/        _\
  `--...__|`-._  _.-'|__...--'
          |    `'    |}%<<<
\AddAnimal[tail-symbol=|,tail-count=1]{crusader}%>>>
{ |
\[T]/}
\AnimalOptions{crusader}{tail-1=|,body-align=c}%<<<
%^^A http://ascii.co.uk/art/knights
\AddAnimal[tail-count=3]{knight}%>>>
{     \
       \   ,-"""-.
        \  | === |
           )  |  (
        .=='\" "/`==.
      .'\   (`:')   /`.
    _/_ |_.-' : `-._|__\_
   <___>'\    :   / `<___>
   /  /   >=======<  /  /
 _/ .'   /  ,-:-.  \/=,'
/ _/    |__/v^v^v\__) \
\(\)     |V^V^V^V^V|\_/
 (\\     \`---|---'/
   \\      \-._|_,-/
    \\      |__|__|
     \\   <___X___>
      \\   \..|../
       \\   \ | /
        \\  /V|V\
         \|/  |  \
          '--' `--`}%<<<
⟨/animals⟩
```

```
 ----------------------------
/                            \
  Who's gonna use it anyway?
\                            /
 ----------------------------
   O
    o   __
      >(' )
        )/
       /(
      /  `----/
      \  ~=- /
    ~^~^~^~^~^~^~^
```

```
 ---------------------------------------
/                                       \
|               Hosted at               |
| https://github.com/Skillmon/ltx_ducksay |
|                it is.                 |
\                                       /
 ---------------------------------------
   \
    \
   __.-._
   '-._"7'
    /'.-c
    |  /T
   _)_/LI
```