Appendix B: Model Functions

```
#' A caching Function
#'
#' Create a model matrix, which is really a list, containing functions to...
#' * move, rotate, calculate, clip, show, ... the model
#'
#' @param model_data numeric vector contents.
#' @keywords caching
#' @examples model <- make_model( list( theta=-90, phi=-2, fov=25, zoom=1 ) )
#' @export

#+ cache_model, echo = TRUE
make_model <- function( model_data ){
  cache       <- NULL                                        # Begin with an empty model

  cache_it    <- function( model_data ) cache <<- model_data  # Cache the model data

  calculate   <- function( f ) apply( model_data, 2, f )

  # Cut off the top/bottom, front/back, or left/right sides of the model
  clip_at <- function( ax=1, mn=0.3, mx=0.6 ){
    model_data <<- model_data[ model_data[ ,ax ] > mn & model_data[ ,ax ] < mx, ]
  }

  data_length <- function() cache                            # num elements in matrix

  extents     <- function() apply( model_data, 2, function( x ) max( x ) - min( x ) )

  get         <- function() name_axes( model_data )          # Return the model data

  # Clip a thin band out of the middle, and keep that portion that was removed
  get_band        <- function( ax, ctr, thickness ){
    mn <- ctr - 0.5 * thickness
    mx <- ctr + 0.5 * thickness
    model_data <<- model_data[ model_data[ ,ax ] > mn & model_data[ ,ax ] < mx, ]
  }

  move_forward_backward <- function( distance ) model_data[ ,3 ] <<- model_data[ ,3 ] + distance
  move_right_left       <- function( distance ) model_data[ ,1 ] <<- model_data[ ,1 ] + distance
  move_up_down          <- function( distance ) model_data[ ,2 ] <<- model_data[ ,2 ] + distance
  rotate_on   <- function( ax, angle ){
   switch( ax
     , x =  model_data <<- rotate3d( model_data, angle * pi / 180, 1, 0, 0 )
     , y =  model_data <<- rotate3d( model_data, angle * pi / 180, 0, 1, 0 )
     , z =  model_data <<- rotate3d( model_data, angle * pi / 180, 0, 0, 1 )
   )
  }
  scale_it    <- function( scale_factor ){
    model_data[ ,1 ] <<- scale_factor * model_data[ ,1 ]
    model_data[ ,2 ] <<- scale_factor * model_data[ ,2 ]
    model_data[ ,3 ] <<- scale_factor * model_data[ ,3 ]
  }

  show         <- function( config, limits=c( -1, 1 ) ){
    clear3d()
    par3d( cex = 2.0 )
    plot3d(
        name_axes( model_data )
      , xlim = limits , ylim = limits , zlim = limits
      , axes = FALSE, box = FALSE
    )
    abclines3d( x = matrix( 0, ncol = 3 ), a = diag( 3 ), col = 'black', lwd = 3 )
    view( config )
    background( 'white' )
  }

  # Return the list of functions
  list(
      cache_it                = cache_it
    , calculate               = calculate
    , clip_at                 = clip_at
    , data_length             = data_length
    , extents                 = extents
    , get                     = get
    , get_band                = get_band
    , move_forward_backward = move_forward_backward
    , move_left_right         = move_left_right
    , move_up_down            = move_up_down
    , rotate_on               = rotate_on
    , show                    = show
    , scale_it                = scale_it
  )
}
```

```
#' A caching Function
#'
#' Create a model, or update it if it exists already.
#'
#' @param model_data numeric vector contents. triple dot.
#' @keywords caching
#' @examples model <- cache_model( model )
#' @export

cache_model <- function( model_data, ... ){
  # Given the data, see if its length has already been determined
  cache <- model_data$data_length()

  # If so...
  if( !is.null( cache )){
    message( "getting cached data" ) # Announce use of cached data
    cache                            # Return the length of the model_data matrix
  }

  # Otherwise...
  data <- model_data$get()           # Get the model_data
  cache <- length( data, ... )       # cache the model_data
  model_data$cache_it( cache )       # Remember the model_data
  cache                              # Return the cached model_data
}

# Examples
# model <- NULL
# model <- make_model( readSTL( filename, plot=FALSE ) )
# model_matrix <- model$get()
# cache_model( model )
# print( model$data_length())
# model$data_length()
# model$move_up( 0.25 )
# model$move_down( 0.25 )
# model$move_left( 0.25 )
# model$move_right( 0.25 )
# model$move_forward( 0.25 )
# model$move_backward( 0.25 )
```

```
## NOT USED
  #  text3d(
  #    matrix(
  #      c(
  #           0.9, 0.2 , 0.2
  #          , 0.2, 0.9 , 0.2
  #          , 0.2, 0.2 , 0.9
  #      )
  #      , ncol = 3
  #    )
  #    , texts = c( 'X', 'Y', 'Z' )
  #  )
```