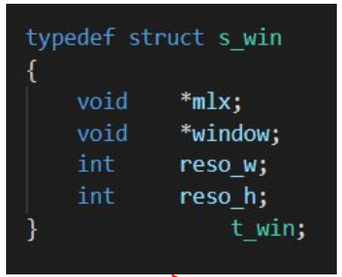
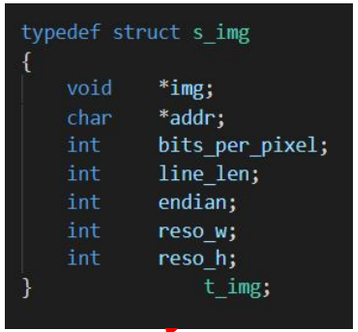
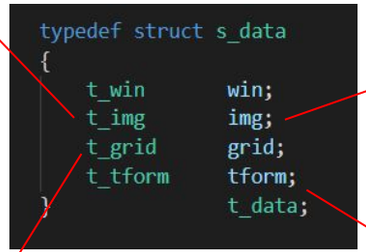


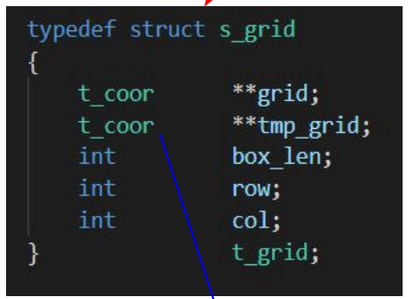
```
typedef struct s_win
{
    void    *mlx;
    void    *window;
    int     reso_w;
    int     reso_h;
}               t_win;
```



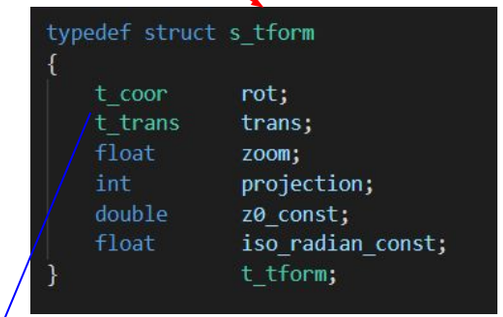
```
typedef struct s_img
{
    void    *img;
    char    *addr;
    int     bits_per_pixel;
    int     line_len;
    int     endian;
    int     reso_w;
    int     reso_h;
}               t_img;
```



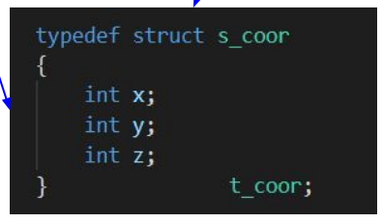
```
typedef struct s_data
{
    t_win       win;
    t_img       img;
    t_grid      grid;
    t_tform     tform;
}               t_data;
```



```
typedef struct s_grid
{
    t_coor      **grid;
    t_coor      **tmp_grid;
    int         box_len;
    int         row;
    int         col;
}               t_grid;
```



```
typedef struct s_tform
{
    t_coor      rot;
    t_trans     trans;
    float       zoom;
    int         projection;
    double      z0_const;
    float       iso_radian_const;
}               t_tform;
```



```
typedef struct s_coor
{
    int x;
    int y;
    int z;
}               t_coor;
```

Initialise struct
i.e. (populate the struct on the left to contain default information before display)

- Initialise grid - reads file and constructs a 2D array, each holding 3D coordinates (x, y, z). Also depending on size of the grid in the file, will adjust the length of each box to accomodate sizes. Returns (-1) and stops function from running if (i) file not found, (ii) grid is invalid.
- Initialise mlx, window and image (t_win & t_img) with preset resolutions (RESO_X and RESO_Y) in header (libfdf.h)
- Initialise transformations, i.e.
  - Defaults rotation (i) angle, (ii) zoom, (iii) translation, (iv)projection
  - Also presets constants to be used by functions later

Transformations of points of grid in t_grid

For each point in the grid,
(i) Applies **rotation**, using 3D rotation matrix for x, y and z axis
(ii) Applies **zoom** by magnifying coordinate position
(iii) Applies isometric **projection** (or perspective projection for bonuses)
(iv) Applies **translation** by moving the pixels
(v) Centers image (i.e. convert from quadrant graph to what computer reads from top left to bottom right)

Out-putting image

(i) Clean image (i.e. erasing everything from screen)
(ii) Print grid lines (using Bresenham's line algorithm)
(iii) Output_grid
(iv) mlx_put_image_to_window

Others
- Needed to use key presses, via mlx hooks
- for rotation, constructed a matrix struct to store values in floats
- colors can be changed
- freeing memory needs to be considered as well, working alongside the when ESC key is pressed as above

Improvements
- Consider making multiplication of rotation matrices a single line instead of being stacked on top of each other
- Consider converting drawline to using ints instead of floats for lesser processing time
- Capturing color gradient next time when drawing lines
- Also incorporate colors in file to have them updated into the t_coor struct (as **unsigned int**?)