main.h

```
#ifndef _MAIN_H
#define _MAIN_H

/* ------------ GLOBAL CONSTANTS, STRUCTS AND VARIABLES ----------------- */

/* Setting global constants and initialising Allegro's stock display */
#define DISPLAY_W 960
#define DISPLAY_H 720

extern ALLEGRO_DISPLAY * disp;

/* Initialising audio functionality for allegro */
extern ALLEGRO_SAMPLE * song;
extern ALLEGRO_SAMPLE_INSTANCE * songInstance;

/* Declaring global constants and initialising all possible key commands
 * for the keyboard logic configuration */
#define KEY_SEEN 1
#define KEY_RELEASED 2

extern unsigned char key [ ALLEGRO_KEY_MAX ];

/* Initialising global variables and flags for the game
 * logic configuration */

extern int frame;
extern int hold_frame;
extern bool done;
extern bool redraw;
extern bool menu;
extern bool music;

/* Declaring global constants for the physics of the game */
#define PI 3.142857
#define G 6.67e-11
#define MOMENTUM_MAX 1e31
#define MOMENTUM_MIN -1e31

/* Initialising Allegro's stock font functionality display and a global
 * variable for incrementing the player's score during the game */
extern ALLEGRO_FONT * font;
extern long score_display;

/* Declaring global constants for the arrow object and declaring the
 * arrow as a global struct */
#define ARROW_SPEED 2

typedef struct arrow
{
        int r, g, b;               /* the colour parameters of the arrow */
        double x, y;               /* the x and y posiitons of the arrow */
        double theta, mag;         /* the angle and magnitude of the arrow */
} arrow_t;

extern arrow_t arrow;

/* Declaring global constants for the stars and declaring star as
 * a global array of structs */
#define ST_N ( ( DISPLAY_W / 2 ) - 1 )   /* number of stars */

typedef struct star
{
        double y;
        double speed;
} star_t;
extern star_t star [ ST_N ];

/* Declaring global constants for the anti-token and declaring
 * the anti-token as a global array of structs */
#define AT_R 15                  /* radius of the anti-token */
#define AT_N 10                  /* number of anti-tokens */

typedef struct anti_token
{
        int r, g, b;      /* colour parameters of the anti-token */
```

```
76          double kg, stiff; /* mass and stiffness of the anti-token's spring */
77          bool visible;     /* whether the anti-tken is visible or not */
78          double x, y;
79          double force_x, force_y;
80          double momentum_x, momentum_y;
81  } anti_token_t;
82  extern anti_token_t anti_token [ AT_N ];
83
84  /* Declaring global constants for the blue_token object and declaring the
85   * blue token as a global struct */
86  #define BT_R 15                   /* radius of the blue token */
87
88  typedef struct blue_token
89  {
90          int r, g, b;              /* colour parameters of the blue token */
91          double kg;
92          double x, y;
93          double x_hat, y_hat;      /* unit vectors of the blue token */
94          double force_x, force_y;
95          double momentum_x, momentum_y;
96          bool live;                /* state of the blue token */
97  } blue_token_t;
98  extern blue_token_t b_token;
99
100 /* Declaring global constants for the green token and declaring the
101  * green token as a global struct */
102 #define GT_R 15                   /* radius of the green token */
103
104 typedef struct green_token
105 {
106         int r, g, b;              /* colour parameters of the green token */
107         double kg;
108         double x, y;
109         double x_hat, y_hat;      /* unit vector of the green token */
110         double force_x, force_y;
111         double  momentum_x, momentum_y;
112 } green_token_t;
113 extern green_token_t g_token;
114
115 /* Declaring global constants for yellow token and declaring the yellow
116  * token as a global struct */
117 #define YT_R 45                   /* radius of the yellow token */
118
119 typedef struct yellow_token
120 {
121         int r, g, b;              /* colour parameters of the yellow token */
122         double x, y;
123 } yellow_token_t;
124 extern yellow_token_t y_token;
125
126
127 /* ---- FUNCTION PROTOTYPES  ------------------------------------------ */
128
129 void must_init ( bool, const char * );
130 int between ( int, int );
131 double between_f ( double, double );
132 double vector_mag ( double, double );
133 bool collide ( int, int,
134                int, int,
135                int, int,
136                int, int );
137 double boundary ( double value );
138
139 void disp_init ( void );
140 void audio_init ( void );
141 void sample_trigger ( void );
142 void audio_destroy ( void );
143 void keyboard_init ( void );
144 void keyboard_update ( ALLEGRO_EVENT * );
145 void score_init ( void );
146 void score_draw ( void );
147 void instruction_draw ( void );
148 void anti_token_init ( void );
149 void anti_token_update ( void );
150 void anti_token_draw ( void );
```

```
151 void blue_token_init ( void );
152 void blue_token_trigger ( void );
153 void blue_token_update ( void );
154 void blue_token_draw ( void );
155 void green_token_init ( void );
156 void green_token_update ( void );
157 void green_token_draw ( void );
158 void yellow_token_init ( void );
159 void yellow_token_update ( void );
160 void yellow_token_draw ( void );
161 void arrow_init ( void );
162 void arrow_update ( void );
163 void arrow_draw ( void );
164 void stars_init ( void );
165 void stars_update ( void );
166 void stars_draw ( void );
167
168 #endif /* _MAIN_H */
```