```
// generated by Fast Light User Interface Designer (fluid) version 1.0302

#include "scrolling.h"
/**
```

1 Specification

This program is a game set in a space theme. The user controls a spaceship that shoots at enemy space craft. The user will start at the main menu which gives the user the ability to START, QUIT or go to the OPTIONS. In the options the user has the ability to customize the game by changing sound and background via the zone option as well as changing the difficulty. While the player is alive he gains movement and shoots the multiple existing UFOs. When one is killed by the laser shooting, the player gains score points shown top right game screen. A sounds will occur on various events such as collision and the boss spawning. When the player's ship runs into an object, the game is over. The game borrows a mechanic from the famous Windows 3.1 game SkiFree where no matter what the Yeti always eats the user. Similar to SkiFree, the boss that spawns is unkillable and always collides with the user's space ship.

```
*/
//
/**
```

2 Analysis

Inputs:

- Up key
- Down key
- Left key
- Right key
- Spacebar

Process:

- Pressing "Play" will start the game
- Pressing "Quit" will close the game
- "Up" will increase the Y coordinate of the ship's position.
- "Down" will decrease the Y coordinate of the ship's position.
- "Left" will decrease the X coordinate of the ship's position.
- "Right" will increase the X coordinate of the ship's position.
- Once the game is started the program will spawn enemies and move them at random.
- The SPACEBAR will initiate the firing function by setting a boolean to true, which is always set to false by default, which after shooting will be reset to false.
- A soundtrack will play upon opening the program, and another sound will be played when the user triggers the firing boolean (e.g. sound effects for the laser or background music for the game itself).

Outputs:

- Main menu window with stylized game title and menu buttons
- Game window
- User controlled ship movement
- Unpredictable movement of enemy UFOs
- Animation of background image

- Animation of ship (2 frames)
- Animation of laser
- Score in an FL_Value_Output box
- Sound effect for triggering of laser
- Sound effect for collision of player with UFOs
- Sound effect for collision of laser with UFOs
- Soundtrack

*/
//
/**

3 Design

Menu

1. Create two boxes
   (a) menu background (menubg)
   (b) title
2. Create buttons for the options
   (a) play
      i. When pressed, make sure the menu is hidden by using menu->hide()
      ii. Make the game shown when pressed with the game->show()
      iii. Hide the options too with options->hide()
      iv. Reset the position of the game with ufo_reset
      v. change the game's difficulty and zone with get_difficulty and get_zone
      vi. use an Fl::add_timeout to reset the score

vii. quit
    A. Make game, game, and options hide with ->hide()
viii. option
    A. Make game, and menu hide with ->hide()
    B. Make options show with options->show()

Options

(a) Two boxes
    i. background (bg)
    ii. options bg (optionbg)

(b) Create two groups for different options
    i. zone
        A. balmora
        B. voss
        C. tython
    ii. (difficulty)
        A. easy
        B. medium
        C. hard

(c) Add a button called "back to main"

Game Window

(a) Create a class that inherits from Fl_Double_Window.

(b) Play background music

(c) Create a (Fl_Box) in the window that is the same size as the background image

(d) Create a callback function that retrieves the background image to animate the background. Place the image in the appropriate box tell it to move the background leftwards to give the impression to the user that they are moving forward inside the function body. Make use of the repeat_timeout function to achieve this.

(e) Create multiple objects using (Fl_Box) again from the class in main Make one for: the background (bg), ship, laser, ufo, and boss.

(f) Create a declaration so that the PDF can read smoothly appropriate images file (ex. GIF, PNG)

(g) Load the boxes images using the (load_images) function.

(h) Create another callback function for all the objects into boxes

(i) Create an IF statement within the callback to cycle the images located within your computer

(j) To cycle the images, use a the timer callback function that changes the time of the images

(k) add a Value_Output that outputs the score when laser hits ufos/boss

  i. Name the Value_Output SCORE

  ii. Make the score visible with SCORE->value

(l) Create a Code box to hide the laser

  i. Add 3 Fl::add_timeouts to delay the moving background, ship, and moving menu's background

(m) Create another Code box to add another Fl::add_timeout for the ufos and, to hide the bossfight.

(n) Add the irrklang library to include background music.

Keyboard Functions

(a) Create a class

  i. Create a new function within the class to make new parameters

  ii. Inside of the same class, create a function called Box with (int x,int y,int w,int h,const char):Fl_Box(x,y,w,h,t) as parameters.

(b) Create another function called handle with the integer event as a parameter

  i. Create a switch statement that will allow the user to switch to different keyboard controls

ii. Create cases to move the ship and fire the laser (Up, Down, Left, Right keys and spacebar)

(c) Create an IF statement for each keyboard control (ex. Up, Down)

i. Make the parent function redraw with parent->redraw()

Zone

1. Create a function called get_zone()
2. Create a variable named Z and make it equal to 1 by default
   (a) Add an IF statement that allows one choice to be selected according to variable
      i. IF player chooses balmora: z = 3
      ii. ELSE IF player chooses voss: z = 2
      iii. ELSE the player chooses the default z = 1
3. Allow it to print what zone is chosen with a cout statement

UFO Difficulty

1. Create a function called get_difficulty()
2. Create a variable named difficulty and make it equal to 3 by default
   (a) Add an IF statement that allows the difficulty to be defualted to hard when selected according
      i. IF player chooses hard: difficulty = 3
      ii. ELSE IF player chooses medium: difficulty = 2
      iii. ELSE player chooses easy: difficulty = 1

Move Laser function

1. Create an integer variable and name it xoflaser. Shift it up to the top of the program. Create a box called laser inside of the main function to house the laser image

2. Create a new function to move the laser across the window and name it move_laser(void* p) to act on the box named laser created to house the laser image

3. Create code to put into the function. Inside the code, set laser to show and a boolean called shooting to control when the laser is triggered.

4. Create an if statement that says that the laser will be hidden behind the ship if shooting is not true. Create an integer variable and name it speed. Initialize it to 10.

5. Set xoflaser to laser->x()+speed

6. Position the laser using laser->position(xoflaser, laser->y())

7. Tell the function to redraw the laser using the redraw function.

8. Create if and else statements to tell it to show the laser and keep it moving across the window until it exits the boundaries of the window and when and if that happens, tell it to reset the laser back where the ship is using the remove_timeout function and repeat_timeout function

UFO spawning

1. Create 6 copies of the UFO spawning code

2. Create a Code box

3. Inside, Create an IF statement that checks laser's collision with the UFO

   (a) Create a nested IF statement so the UFO's position in the X coordinate stays the same, while the Y coordinate keeps the UFO random

      i. then allow the UFO to show

      ii. Create a variable named t = 0

      iii. Create an equation: t = t + 100

      iv. Allow the score to be shown with t

v. Add music when the ufo dies

(b) Add an ELSE statement so it checks the UFO so it checks when the it collides with the ship

i. Allow the score to reset

ii. Play a sound when both the ship and UFO collide

iii. Make the game hide using game->hide()

iv. Make the menu show using menu -> show()

(c) Add an ELSE IF statement that checks the UFO's X position and width if greater than 0

i. Move the UFO's position randomly

i. Add an ELSE statement so that the ufo's position resets

A. Make the UFO show using ufo->show();

(d) Create an Fl::repeat_timeout so that the ufos move 1 out of 130 seconds

Scrolling Background function

1. create Fl_Box inside a window that is equivalent to the background pictures height and width
2. create a static int for the starting x coordinate of the object
3. set the speed to an int
4. update x axis of the background with bg->position
5. redraw the window with bgwin->redraw() function
6. use Fl::repeat_timeout to loop the code every .01 seconds
7. if the x position of the background image exceeds the size of the picture width subtracted by the width of the background window then return it to its original position

Ship images cycling function

1. declare sstream header

2. create a new box thats the same size as the images used for animation

3. create create a callback function that will be used to animate the ship

4. create a box for the ship and use an array to hold the images.

5. load the images in (load_images) function.

6. create a callback function for the ship and invoke it in main using Fl::add_timeout function

7. In the callback cycle through the images

Score

1. Create a function called get_score(void*)

   (a) Add an IF statement that explains that if the score > 3000 it spawns boss

2. Add an Fl::add_timeout that checks for the the boss continuously

3. Add a remove_timeout to cancel the function

4. Print BOSS TIME while playing the boss' music

   (a) ELSE the Fl::repeat_timeout repeats

UFO reset

1. Create a function called move_laser(void*)

   (a) Set the ufo's position to a random height while the x position of the game stays the same by using ufo->show()

   (b) Do this 6 times, while changing the UFO's variable name (ex. ufo2, ufo3, ufo4, etc.)

   (c) Set the boss's position to a certain X coordinate, but a random height

   (d) Set the show's position to (40,210)

Laser reset

1. Make the laser show by using laser->show()
2. Create a static boolean called shooting
   (a) Create an IF statement with a boolean that sets the laser's position to stay hidden behind the ship by default
      i. if the boolean is true, the space bar will shoot and play a sound
3. create an integer named speed equal to 10
   (a) allow xoflaser = laser->x() + speed // meaning it controls the speed
   (b) allow the laser to be redrawn by using laserwin->redraw()

1. Create an IF statement
   (a) Allow if the shooting boolean is false, reset the laser to (0, -100)
   (b) Create another IF statement with an FL::repeat_tmeout as a loop
      i. Make an Fl::remove_timeout function to stop the laser

Boss Spawn

1. Create an IF statement so that the position of the boss' bos meets all the conditions to line up with the ship
2. Create an ELSE IF statement so the score triggers the boss spawn
   (a) Make the boss show by using boss->show()
   (b) Have the boss move 1 out of 30 seconds using the Fl::repeat_timeout function

```
*/
//
/**
```

4 Implementation

```
*/
\\
//================================================
//================= INCLUDE FILES =================
#include <FL/Fl_JPEG_Image.H>
#include <FL/Fl_GIF_Image.h>
#include <FL/Fl_PNG_Image.H>
#include <iostream>
#include <sstream>
#include <ctime>
#include <irrKlang.h>
//================================================
//======== NAME SPACE & GLOBAL VARIABLES ========
using namespace irrklang;
using namespace std;
static ISoundEngine* engine = createIrrKlangDevice();
static const int SHIP_FRAMES = 2;
static Fl_GIF_Image* ship_frames[SHIP_FRAMES];
static int xoflaser;
static int difficulty = 1;
//================================================
//============= KEYBOARD ORIENTATION ============

Box::Box(int x,int y,int w,int h,const char* t):Fl_Box(x,y,w,h,t) {
  printf("Welcome to SpaceScroller!\n");
}

int Box::handle(int event) {
```

```
int r = 0;
static int x1;
static int y1;
switch(event)
{
   case FL_KEYDOWN :
    if(Fl::event_key() == FL_Up)
 {
      position(x(),y()-10); // MOVES SPACESHIP UP
      parent()->redraw();
      r=1; printf("1\n"); break;
}

    if(Fl::event_key() == 32) {
        printf("space\n");
        r=1;
        Fl::add_timeout(0.1, move_laser, game); // FIRES THE LASER
      }
    if(Fl::event_key() == FL_Down)
  {
      position(x(),y()+10); // // MOVES SPACESHIP DOWN
      parent()->redraw();
      r=1; printf("1\n"); break;
}
 if(Fl::event_key() == FL_Right)
  {
      position(x()+10,y()); // MOVES SPACESHIP RIGHT
      parent()->redraw();
      r=1; printf("1\n"); break;
}
 if(Fl::event_key() == FL_Left)
  {
      position(x()-10,y()+6); // MOVES SPACESHIP LEFT
      parent()->redraw();
      r=1; printf("1\n"); break;
}

//printf("z\n");
```

```
  //xoflaser = 228;
  //cout << "here";
         break;
     case FL_FOCUS :
        r = 1; printf("x\n"); break;

     case FL_UNFOCUS :
                r = 1; printf("y\n"); break;
  }

  return r;
}
//================================================
//================= MAIN FUNCTION ================

Fl_Double_Window *options=(Fl_Double_Window *)0;

Fl_Box *optionbg=(Fl_Box *)0;

Fl_Box *backdrop=(Fl_Box *)0;

Fl_Group *zone=(Fl_Group *)0;

Fl_Round_Button *balmora=(Fl_Round_Button *)0;

Fl_Round_Button *voss=(Fl_Round_Button *)0;

Fl_Round_Button *tython=(Fl_Round_Button *)0;

Fl_Group *DIFFICULTY=(Fl_Group *)0;

Fl_Round_Button *easy=(Fl_Round_Button *)0;

Fl_Round_Button *medium=(Fl_Round_Button *)0;

Fl_Round_Button *hard=(Fl_Round_Button *)0;

Fl_Double_Window *game=(Fl_Double_Window *)0;
```

```
Fl_Box *bg=(Fl_Box *)0;

Box *ship=(Box *)0;

Box *laser=(Box *)0;

Box *ufo=(Box *)0;

Box *ufo2=(Box *)0;

Box *ufo3=(Box *)0;

Box *ufo4=(Box *)0;

Box *ufo5=(Box *)0;

Box *ufo6=(Box *)0;

Box *boss=(Box *)0;

Fl_Value_Output *SCORE=(Fl_Value_Output *)0;

static void cb_SCORE(Fl_Value_Output*, void*) {
  SCORE->value();
}

Fl_Double_Window *menu=(Fl_Double_Window *)0;

Fl_Box *menubg=(Fl_Box *)0;

Fl_Button *play=(Fl_Button *)0;

static void cb_play(Fl_Button*, void*) {
  //this gides the menu and shows the game to start the game
game->show();
menu->hide();
options->hide();
```

```
ufo_reset(); //resets position of objects;
}

Fl_Button *quit=(Fl_Button *)0;

static void cb_quit(Fl_Button*, void*) {
  //the program quits once you hide all windows
game->hide();
menu->hide();
options->hide();
}

Fl_Button *option=(Fl_Button *)0;

static void cb_option(Fl_Button*, void*) {
  game->hide();
menu->hide();
options->show();
}

Fl_Box *title=(Fl_Box *)0;

int main(int argc, char **argv) {
  { options = new Fl_Double_Window(640, 480);
    { Fl_Box* o = optionbg = new Fl_Box(0, 0, 640, 480);
      o->image( new Fl_JPEG_Image("pictures/optionbg.jpg") );
    } // Fl_Box* optionbg
    { backdrop = new Fl_Box(191, 140, 258, 122);
      backdrop->box(FL_FLAT_BOX);
      backdrop->color((Fl_Color)55);
    } // Fl_Box* backdrop
    { zone = new Fl_Group(313, 177, 95, 68, "Combat Zone");
      { balmora = new Fl_Round_Button(325, 205, 64, 15, "Balmora");
        balmora->down_box(FL_ROUND_DOWN_BOX);
      } // Fl_Round_Button* balmora
      { voss = new Fl_Round_Button(325, 229, 64, 15, "Voss");
        voss->down_box(FL_ROUND_DOWN_BOX);
      } // Fl_Round_Button* voss
```

```
    { tython = new Fl_Round_Button(326, 182, 64, 15, "Tython");
      tython->down_box(FL_ROUND_DOWN_BOX);
    } // Fl_Round_Button* tython
    zone->end();
  } // Fl_Group* zone
  { DIFFICULTY = new Fl_Group(209, 178, 89, 68, "Difficulty");
    { easy = new Fl_Round_Button(231, 183, 64, 15, "easy");
      easy->down_box(FL_ROUND_DOWN_BOX);
    } // Fl_Round_Button* easy
    { medium = new Fl_Round_Button(231, 205, 64, 15, "medium");
      medium->down_box(FL_ROUND_DOWN_BOX);
    } // Fl_Round_Button* medium
    { hard = new Fl_Round_Button(231, 229, 64, 15, "hard");
      hard->down_box(FL_ROUND_DOWN_BOX);
    } // Fl_Round_Button* hard
    DIFFICULTY->end();
  } // Fl_Group* DIFFICULTY
  options->end();
} // Fl_Double_Window* options
{ game = new Fl_Double_Window(639, 478, "PewPew");
  { Fl_Box* o = bg = new Fl_Box(0, 0, 10321, 480);
    o->image( new Fl_JPEG_Image("pictures/background.jpg") );
  } // Fl_Box* bg
  { ship = new Box(40, 210, 127, 35);
    ship->box(FL_NO_BOX);
    ship->color(FL_BACKGROUND_COLOR);
    ship->selection_color(FL_BACKGROUND_COLOR);
    ship->labeltype(FL_NORMAL_LABEL);
    ship->labelfont(0);
    ship->labelsize(14);
    ship->labelcolor(FL_FOREGROUND_COLOR);
    ship->align(Fl_Align(FL_ALIGN_CENTER));
    ship->when(FL_WHEN_RELEASE);
  } // Box* ship
  { Box* o = laser = new Box(170, 241, 128, 9);
    laser->box(FL_NO_BOX);
    laser->color(FL_BACKGROUND_COLOR);
    laser->selection_color(FL_BACKGROUND_COLOR);
```

```
  laser->labeltype(FL_NORMAL_LABEL);
  laser->labelfont(0);
  laser->labelsize(14);
  laser->labelcolor((Fl_Color)2);
  laser->align(Fl_Align(FL_ALIGN_CENTER));
  laser->when(FL_WHEN_RELEASE);
  o->image( new Fl_GIF_Image("laser2.gif") );
} // Box* laser
{ Box* o = ufo = new Box(585, 43, 40, 42);
  ufo->box(FL_NO_BOX);
  ufo->color(FL_BACKGROUND_COLOR);
  ufo->selection_color(FL_BACKGROUND_COLOR);
  ufo->labeltype(FL_NORMAL_LABEL);
  ufo->labelfont(0);
  ufo->labelsize(14);
  ufo->labelcolor(FL_FOREGROUND_COLOR);
  ufo->align(Fl_Align(FL_ALIGN_CENTER));
  ufo->when(FL_WHEN_RELEASE);
  o->image( new Fl_PNG_Image("pictures/ufo.png") );
} // Box* ufo
{ Box* o = ufo2 = new Box(580, 98, 40, 42);
  ufo2->box(FL_NO_BOX);
  ufo2->color(FL_BACKGROUND_COLOR);
  ufo2->selection_color(FL_BACKGROUND_COLOR);
  ufo2->labeltype(FL_NORMAL_LABEL);
  ufo2->labelfont(0);
  ufo2->labelsize(14);
  ufo2->labelcolor(FL_FOREGROUND_COLOR);
  ufo2->align(Fl_Align(FL_ALIGN_CENTER));
  ufo2->when(FL_WHEN_RELEASE);
  o->image( new Fl_PNG_Image("pictures/ufo.png") );
} // Box* ufo2
{ Box* o = ufo3 = new Box(560, 153, 40, 42);
  ufo3->box(FL_NO_BOX);
  ufo3->color(FL_BACKGROUND_COLOR);
  ufo3->selection_color(FL_BACKGROUND_COLOR);
  ufo3->labeltype(FL_NORMAL_LABEL);
  ufo3->labelfont(0);
```

```
  ufo3->labelsize(14);
  ufo3->labelcolor(FL_FOREGROUND_COLOR);
  ufo3->align(Fl_Align(FL_ALIGN_CENTER));
  ufo3->when(FL_WHEN_RELEASE);
  o->image( new Fl_PNG_Image("pictures/ufo.png") );
} // Box* ufo3
{ Box* o = ufo4 = new Box(555, 223, 40, 42);
  ufo4->box(FL_NO_BOX);
  ufo4->color(FL_BACKGROUND_COLOR);
  ufo4->selection_color(FL_BACKGROUND_COLOR);
  ufo4->labeltype(FL_NORMAL_LABEL);
  ufo4->labelfont(0);
  ufo4->labelsize(14);
  ufo4->labelcolor(FL_FOREGROUND_COLOR);
  ufo4->align(Fl_Align(FL_ALIGN_CENTER));
  ufo4->when(FL_WHEN_RELEASE);
  o->image( new Fl_PNG_Image("pictures/ufo.png") );
} // Box* ufo4
{ Box* o = ufo5 = new Box(575, 298, 40, 42);
  ufo5->box(FL_NO_BOX);
  ufo5->color(FL_BACKGROUND_COLOR);
  ufo5->selection_color(FL_BACKGROUND_COLOR);
  ufo5->labeltype(FL_NORMAL_LABEL);
  ufo5->labelfont(0);
  ufo5->labelsize(14);
  ufo5->labelcolor(FL_FOREGROUND_COLOR);
  ufo5->align(Fl_Align(FL_ALIGN_CENTER));
  ufo5->when(FL_WHEN_RELEASE);
  o->image( new Fl_PNG_Image("pictures/ufo.png") );
} // Box* ufo5
{ Box* o = ufo6 = new Box(585, 368, 40, 42);
  ufo6->box(FL_NO_BOX);
  ufo6->color(FL_BACKGROUND_COLOR);
  ufo6->selection_color(FL_BACKGROUND_COLOR);
  ufo6->labeltype(FL_NORMAL_LABEL);
  ufo6->labelfont(0);
  ufo6->labelsize(14);
  ufo6->labelcolor(FL_FOREGROUND_COLOR);
```

```
    ufo6->align(Fl_Align(FL_ALIGN_CENTER));
    ufo6->when(FL_WHEN_RELEASE);
    o->image( new Fl_PNG_Image("pictures/ufo.png") );
  } // Box* ufo6
  { Box* o = boss = new Box(636, 133, 332, 196);
    boss->box(FL_NO_BOX);
    boss->color(FL_BACKGROUND_COLOR);
    boss->selection_color(FL_BACKGROUND_COLOR);
    boss->labeltype(FL_NORMAL_LABEL);
    boss->labelfont(0);
    boss->labelsize(14);
    boss->labelcolor(FL_FOREGROUND_COLOR);
    boss->align(Fl_Align(FL_ALIGN_CENTER));
    boss->when(FL_WHEN_RELEASE);
    o->image( new Fl_PNG_Image("pictures/boss.png") );
  } // Box* boss
  { SCORE = new Fl_Value_Output(581, 5, 68, 24, "SCORE: ");
    SCORE->labelfont(5);
    SCORE->labelsize(16);
    SCORE->labelcolor(FL_BACKGROUND2_COLOR);
    SCORE->textfont(4);
    SCORE->callback((Fl_Callback*)cb_SCORE);
  } // Fl_Value_Output* SCORE
  game->end();
} // Fl_Double_Window* game
{ menu = new Fl_Double_Window(640, 480);
  menu->when(FL_WHEN_RELEASE_ALWAYS);
  { Fl_Box* o = menubg = new Fl_Box(0, 0, 10321, 480);
    o->image( new Fl_JPEG_Image("pictures/menubg.jpg") );
  } // Fl_Box* menubg
  { Fl_Button* o = play = new Fl_Button(252, 215, 140, 45);
    play->color(FL_FOREGROUND_COLOR);
    play->selection_color(FL_FOREGROUND_COLOR);
    play->callback((Fl_Callback*)cb_play);
    o->image( new Fl_GIF_Image("pictures/play.gif") );
  } // Fl_Button* play
  { Fl_Button* o = quit = new Fl_Button(252, 375, 140, 45);
    quit->color(FL_FOREGROUND_COLOR);
```

```
      quit->selection_color(FL_FOREGROUND_COLOR);
      quit->callback((Fl_Callback*)cb_quit);
      o->image( new Fl_GIF_Image("pictures/quit.gif") );
    } // Fl_Button* quit
    { option = new Fl_Button(252, 295, 140, 45, "options");
      option->callback((Fl_Callback*)cb_option);
    } // Fl_Button* option
    { Fl_Box* o = title = new Fl_Box(4, 18, 640, 160, " ");
      o->image( new Fl_GIF_Image("pictures/title.gif") );
    } // Fl_Box* title
    menu->end();
  } // Fl_Double_Window* menu
  //timeout functions for animations
  laser->hide();
  load_ship_images();
  Fl::add_timeout(1.0,move_bg, game);
  Fl::add_timeout(1.0, shipanimation);
  Fl::add_timeout(1.0,move_menubg, menu);
  //timeout functions for ufo's
  Fl::add_timeout(2.0,cb_spawn);
  // the soundtrack
  engine->play2D("music.mp3", true);
  menu->show(argc, argv);
  return Fl::run();
}
//================================================
//==================GAME MECHANICS==============

void cb_spawn(void*) {
  //ufo1
  static int x=0;
  static int t=0;

  if(ufo->x()+ufo->w() > laser->x() &&
     ufo->x()< laser->x()+ laser->w() &&
     ufo->y()+ufo->h() > laser->y() &&
     ufo->y()< laser->y()+laser->h())
     //check laser col. with ufo
```

```
      {
      ufo->position(  game->w(), rand()%game->h()/2 );
      ufo->show();

//resets position
      t = t+100;
      SCORE->value(t);

//increment score and show
      engine->play2D("adeath.wav");
      }

else if(ufo->x()+ufo->w() > ship->x() &&
 //check ufo col. with ship
   ufo->x()< ship->x()+ ship->w() &&
   ufo->y()+ufo->h() > ship->y() &&
   ufo->y()< ship->y()+ship->h())
      {
      t=0;
      SCORE->value(t);
              //reset score
      engine->play2D("pdeath.wav");
      ufo_reset();
              //reset ufo positions
      game->hide();
      menu->show();
      }

else if(ufo->x() + ufo->w() > 0)
      //if UFO is onscreen
      {
      ufo->position(ufo->x()-1 ,ufo->y()+rand()%5-2);
      //move UFO
      }



else
```

```
      //if ufo is offscreen
      {
      ufo->position(  game->w(), rand()%game->h()/2 );
      //reset position
      ufo->show();
      }

Fl::repeat_timeout(1.0/130,cb_spawn);
      //repeat 130 times a second
//ufo2

if(ufo2->x()+ufo2->w() > laser->x() &&
   ufo2->x()< laser->x()+ laser->w() &&
   ufo2->y()+ufo2->h() > laser->y() &&
   ufo2->y()< laser->y()+laser->h())

   // check laser col. with ufo
      {
      ufo2->position(  game->w(), rand()%game->h()/2 );
      ufo2->show();
       //resets position
      t = t+100;
      SCORE->value(t);
       //increment score and show
      engine->play2D("adeath.wav");
      }

else if(ufo2->x()+ufo2->w() > ship->x() &&
//check ufo col. with ship
   ufo2->x()< ship->x()+ ship->w() &&
   ufo2->y()+ufo->h() > ship->y() &&
   ufo2->y()< ship->y()+ship->h())
      {
      t=0;
      SCORE->value(t);
        //reset score
              engine->play2D("pdeath.wav");
      ufo_reset();
```

```
       //reset ufo positions
      game->hide();

      menu->show();
      }

else if(ufo2->x() + ufo2->w() > 0)
  //if UFO is onscreen
      {
      ufo2->position(ufo2->x()-1 ,ufo2->y()+rand()%5-3);
      //move UFO
      }



else

 //if ufo is offscreen
      {
      ufo2->position(  game->w(), rand()%game->h()/2 );
       //reset position
      ufo2->show();
      }
//ufo3

if(ufo3->x()+ufo3->w() > laser->x() &&
   ufo3->x()< laser->x()+ laser->w() &&
   ufo3->y()+ufo3->h() > laser->y() &&
   ufo3->y()< laser->y()+laser->h())
     //check laser col. with ufo
      {
      ufo3->position(  game->w(), rand()%game->h()/2 );
      ufo3->show();
       //resets position
      t = t+100;

      SCORE->value(t);
           //increment score and show
```

```
      engine->play2D("adeath.wav");
      }

else if(ufo3->x()+ufo3->w() > ship->x() &&
//check ufo col. with ship
   ufo3->x()< ship->x()+ ship->w() &&
   ufo3->y()+ufo->h() > ship->y() &&
   ufo3->y()< ship->y()+ship->h())
      {
      t=0;
      SCORE->value(t);

      //reset score
              engine->play2D("pdeath.wav");
      ufo_reset();
       //reset ufo positions
      game->hide();
      menu->show();
      }

else if(ufo3->x() + ufo3->w() > 0)
 //if UFO is onscreen
      {
      ufo3->position(ufo3->x()-1 ,ufo3->y()+rand()%5-3);
       //move UFO
      }



else
//if ufo is offscreen
      {
      ufo3->position(  game->w(), rand()%game->h()/2 );
       //reset position
      ufo3->show();
      }
//ufo4
```

```
if(ufo4->x()+ufo4->w() > laser->x() &&
   ufo4->x()< laser->x()+ laser->w() &&
   ufo4->y()+ufo4->h() > laser->y() &&
   ufo4->y()< laser->y()+laser->h())

   //check laser col. with ufo
      {
      ufo4->position(  game->w(), rand()%game->h()/2 );
      ufo4->show();

      //resets position
      t = t+100;
      SCORE->value(t);
       //increment score and show
      engine->play2D("adeath.wav");
      }

else if(ufo4->x()+ufo4->w() > ship->x() &&
//check ufo col. with ship
   ufo4->x()< ship->x()+ ship->w() &&
   ufo4->y()+ufo->h() > ship->y() &&
   ufo4->y()< ship->y()+ship->h())
      {
      t=0;
      SCORE->value(t);
       //reset score
              engine->play2D("pdeath.wav");
      ufo_reset();
      //reset ufo positions
      game->hide();
      menu->show();
      }

else if(ufo4->x() + ufo4->w() > 0)
 //if UFO is onscreen
      {
      ufo4->position(ufo4->x()-1 ,ufo4->y()+rand()%5-2);
      //move UFO
```

```
      }



else
  //if ufo is offscreen
      {
      ufo4->position(  game->w(), rand()%game->h()/2 );
       //reset position
      ufo4->show();
      }
//ufo5

if(ufo5->x()+ufo5->w() > laser->x() &&
   ufo5->x()< laser->x()+ laser->w() &&
   ufo5->y()+ufo5->h() > laser->y() &&
   ufo5->y()< laser->y()+laser->h())
    //check laser col. with ufo
      {
      ufo5->position(  game->w(), rand()%game->h()/2 );
      ufo5->show();
       //resets position
      t = t+100;
      SCORE->value(t);
       //increment score and show
      engine->play2D("adeath.wav");
      }

else if(ufo5->x()+ufo5->w() > ship->x() &&
//check ufo col. with ship
   ufo5->x()< ship->x()+ ship->w() &&
   ufo5->y()+ufo->h() > ship->y() &&
   ufo5->y()< ship->y()+ship->h())
      {
      t=0;
      SCORE->value(t);
      //reset score
      engine->play2D("pdeath.wav");
```

```
      ufo_reset();
      //reset ufo positions
      game->hide();
      menu->show();
      }

else if(ufo5->x() + ufo5->w() > 0)
//if UFO is onscreen
      {
      ufo5->position(ufo5->x()-1 ,ufo5->y()+rand()%5-2);
       //move UFO
      }



else
 //if ufo is offscreen
      {
      ufo5->position(  game->w(), rand()%game->h()/2 );
      //reset position
      ufo5->show();
      }
//ufo6

if(ufo6->x()+ufo6->w() > laser->x() &&
   ufo6->x()< laser->x()+ laser->w() &&
   ufo6->y()+ufo6->h() > laser->y() &&
   ufo6->y()< laser->y()+laser->h())
    //check laser col. with ufo
      {
      ufo6->position(  game->w(), rand()%game->h()/2 );
      ufo6->show();
      //resets position
      t = t+100;
      SCORE->value(t);
      //resets position
      engine->play2D("adeath.wav");
      }
```

```
else if(ufo6->x()+ufo6->w() > ship->x() &&
//check ufo col. with ship
   ufo6->x()< ship->x()+ ship->w() &&
   ufo6->y()+ufo->h() > ship->y() &&
   ufo6->y()< ship->y()+ship->h())
      {
      t=0;
      SCORE->value(t);

      //reset score
        engine->play2D("pdeath.wav");
      ufo_reset();
      //reset ufo positions
      game->hide();
      menu->show();
      }

else if(ufo6->x() + ufo6->w() > 0)
 //if UFO is onscreen
      {
      ufo6->position(ufo6->x()-1 ,ufo6->y()+rand()%5-2);
      //move UFO
      }




else
//if ufo is offscreen
      {
      ufo6->position(  game->w(), rand()%game->h()/2 );
       //reset position
      ufo6->show();
      }
//boss

boss->hide();
```

```
  if(boss->x()+boss->w() > ship->x() &&
     boss->x()< ship->x()+ ship->w() &&
     boss->y()+boss->h() > ship->y() &&
     boss->y()< ship->y()+ship->h())
        {
        t=0;
        SCORE->value(t);
        engine->play2D("pdeath.wav");
        ufo_reset();
        game->hide();
        menu->show();
        }

  else if((boss->x() + boss->w() > 0) &&
        (t > 500) ) //<--- score that triggers boss spawn
        {
        boss->show();
        boss->position(boss->x()-1  ,ship->y()-100);
                //move boss
        }
}

void move_laser(void* p) {
  laser->show();
  static bool shooting = true;

  Fl_Double_Window* laserwin = static_cast<Fl_Double_Window*>(p);

  //boolean defaulted to true and goes false then immidiately back to true
  //to reset the lasers position (x,y)
  if ( !shooting ) {
        laser->position( ship->x()+.5*ship->w(), ship->y()+.6*ship->h() );
        shooting = true; engine->play2D("laser.wav");
  }

  // this calculates how fast the object moves
  int speed = 10;
```

```
  // this controls movement,
  // by position (-speed moves left, +speed moves right)
  xoflaser = laser->x()+speed;

  // this positions my object,
  // note that its (x,y) the y for my object is always 0 so
  // its only x i calculate
  laser->position(xoflaser, laser->y());
  // redraws the bg window so that it can display our updated
  // coordinates for our object
  laserwin->redraw();

  // this code removes the laser by stopping it
  //behind the ship when it exceeds the window width
  if ( laser->x() > game->w() ) {
        shooting = false;
        laser->position(0, -100);
        Fl::remove_timeout(move_laser);
  }

  if ( shooting ) {
        // treat Fl::repeat_timeout as a loop
        Fl::repeat_timeout(0.01,move_laser,laserwin);
  }
}

void ufo_reset() {
  //resets positions of all ships and UFOs
  //triggered upon death
  ufo->position(  game->w(), rand()%game->h() );
  ufo->show();

  ufo2->position(  game->w(), rand()%game->h());
  ufo2->show();

  ufo3->position(  game->w(), rand()%game->h() );
  ufo3->show();
```

```
  ufo4->position(  game->w(), rand()%game->h() );
  ufo4->show();

  ufo5->position(  game->w(), rand()%game->h() );
  ufo5->show();

  ufo6->position(  game->w(), rand()%game->h() );
  ufo6->show();

  boss->position(  640, rand()%game->h() );
  boss->show();

  ship->position(40,210);
}
//================================================
//================== ANIMATIONS ================

void shipanimation(void*) {
  static int icounter = 0;
  //set to 0 because first image is ship0.gif
  ship->image(*ship_frames[icounter]);
   //retrieves one of the gifs, starting with ship0.gif
  icounter = (icounter+1)%SHIP_FRAMES;

  //this updates the counter - 0,1,0,1,0,1

  ship->parent()->redraw();
   //this draws the updated image

  Fl::repeat_timeout(.15, shipanimation);
   //after this function is complete recall
   //the function shipanimation every .15 seconds.
}

void load_ship_images() {
  for (int i = 0; i < SHIP_FRAMES; i++)
  //ship frames is 2 set as a global constant  (does it twice)
  {
```

```
  ostringstream oss;
  //opening a string stream buffer to read from the folder
  oss << i;
  //pulls ship 0 - (i is initialized to 0 in the for loop)
  string s = "pictures/ship/ship" + oss.str() + ".gif";
  // stores ship0.gif into the string sgif
  ship_frames[i] = new Fl_GIF_Image( s.c_str() );
   //translates the gif file from that directory into
   //the fluid method Fl_GIF_Image and stores it into the ship_frames array
  }
}

void move_bg(void* p) {
  // scrolling bg
  //call back function is receiving pointer
  // to void therefore convert it to its true type
  //so we can use it
  Fl_Double_Window* bgwin = static_cast<Fl_Double_Window*>(p);

  static int xofbg = 0;
  //this initiates the x starting x position of the object
  int speed = 5;
  //this calculates how fast the object moves
  xofbg = xofbg-speed;
  //this controls movement - moves left, + moves right

  const int picwid = 10321;
  if ( xofbg < -( picwid-bgwin->w() ) )
  //this is the condition met of the object
      xofbg = 0;
  //this brings "Fl_box bg" back to its starting coordinate

  bg->position(xofbg,0);
   //this positions my object,
  bgwin->redraw();
  //redraws the bg window so that it can
  // display our updated coordinates for our object
```

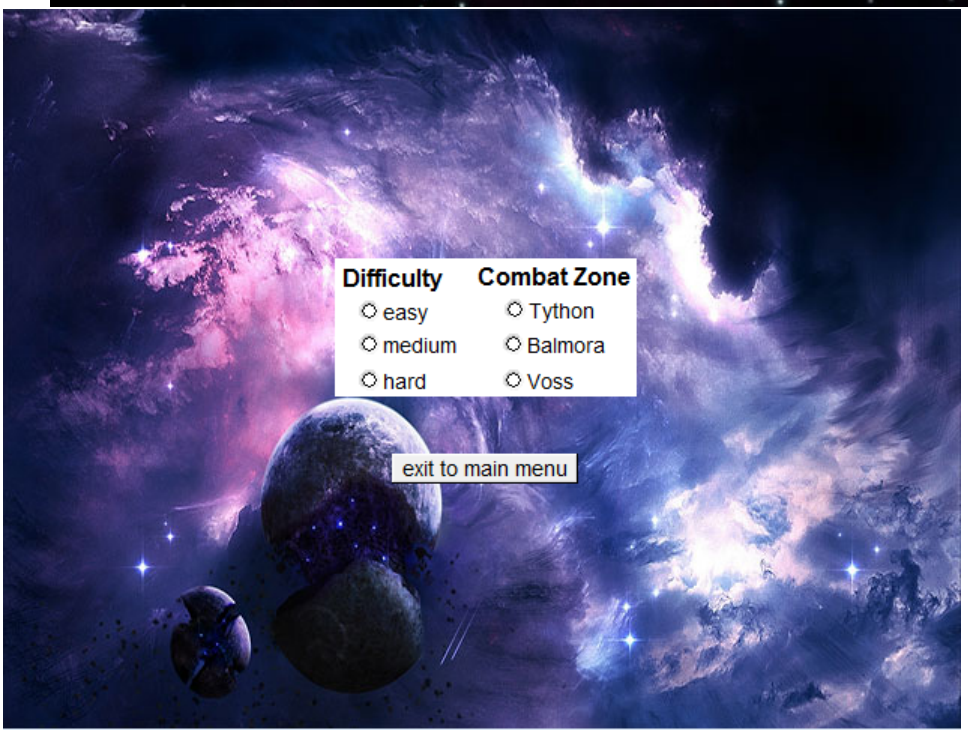
```
  //std::cout << "(" << bg->x() << "," << bg->y() << ")\n";
  //uncomment to output x,y coordinates
  Fl::repeat_timeout(0.01,move_bg,bgwin);
   // treat Fl::repeat_timeout as a loop
}

void move_menubg(void* p) {
  // scrolling bg
  //call back function is
  //receiving pointer to void therefore convert
  // it to its true type so we can use it
  Fl_Double_Window* bgwin = static_cast<Fl_Double_Window*>(p);

  static int xofbg = 0;
  //this initiates the x starting x position of the object
  int speed = 10;
   //this calculates how fast the object moves
  xofbg = xofbg-speed;
   //this controls movement,
   // by position (-speed moves left, +speed moves right)

  const int picwid = 10321;
  if ( xofbg+menubg->w() < bgwin->w() ) xofbg = 0;

  menubg->position(xofbg,0);
  //this positions my object, note that its (x,y) the y
  // for the object is always 0 so its only x i calculate
  bgwin->redraw();
   //redraws the bg window so that it can display our
   // updated coordinates for our object

  //std::cout << "(" << bg->x() << "," << bg->y() << ")\n";
   //this displays the outputs of my
  // object in my msys (note that you have to flag this in your fltk-config
  Fl::repeat_timeout(0.01,move_menubg,bgwin);
   // treat Fl::repeat_timeout as a loop
}
/**
```

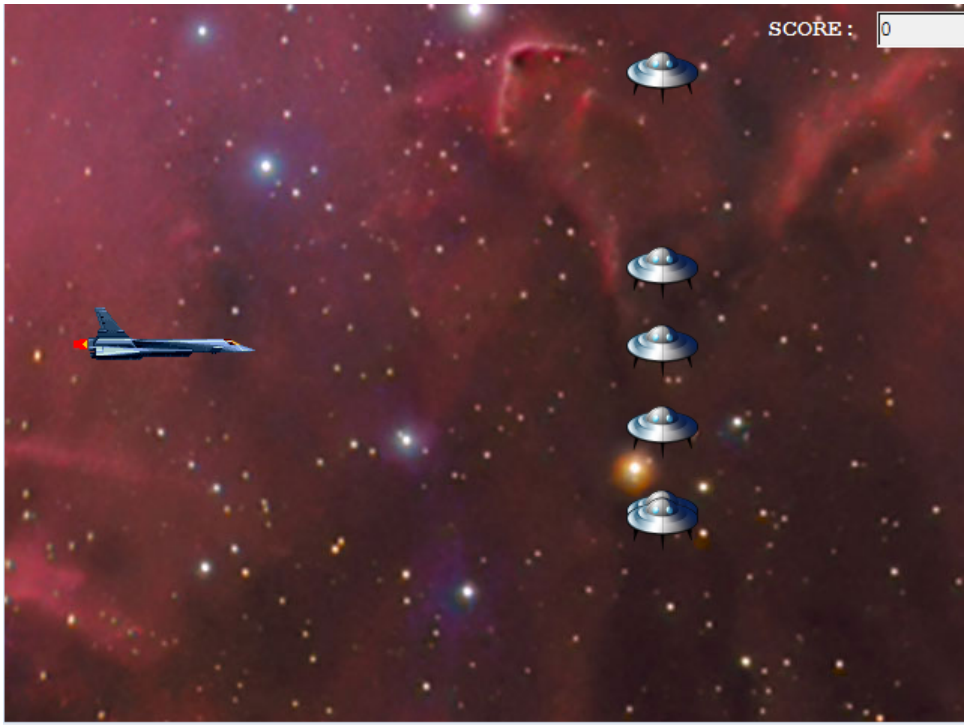
5 Test

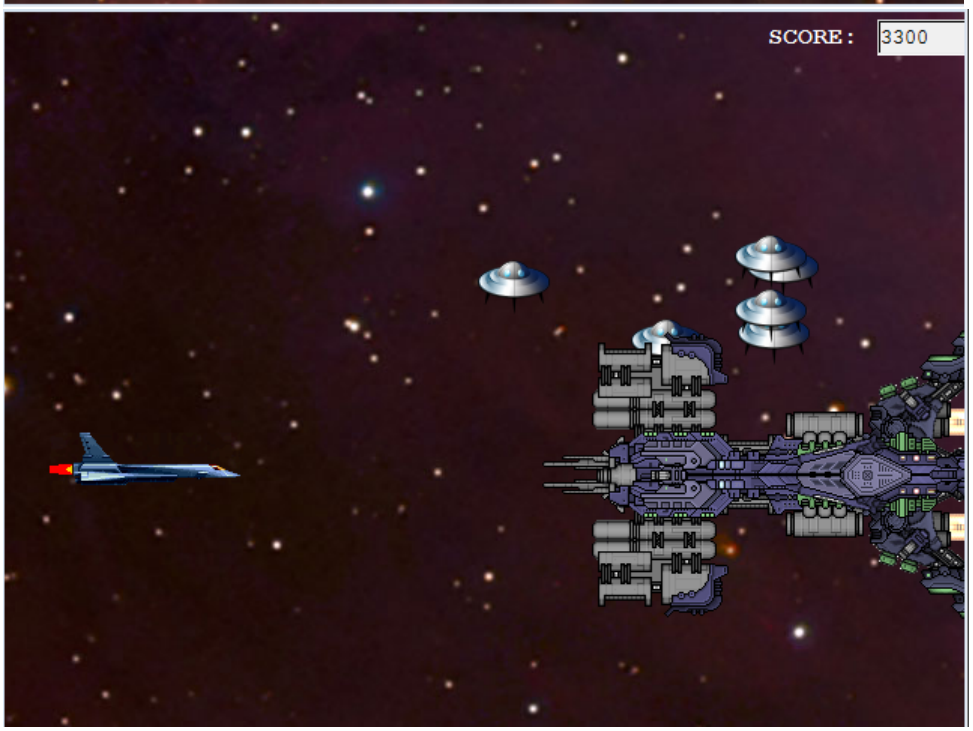
Excpected Result

Actual Result
Video
Screenshots


SPACE
SCROLLER
PLAY
OPTIONS
QUIT


Difficulty
easy
medium
hard
Combat Zone
Tython
Balmora
Voss
exit to main menu


SCORE :
0


SCORE :
3300


0
SCORE :
0

```
    }

*/
//
```