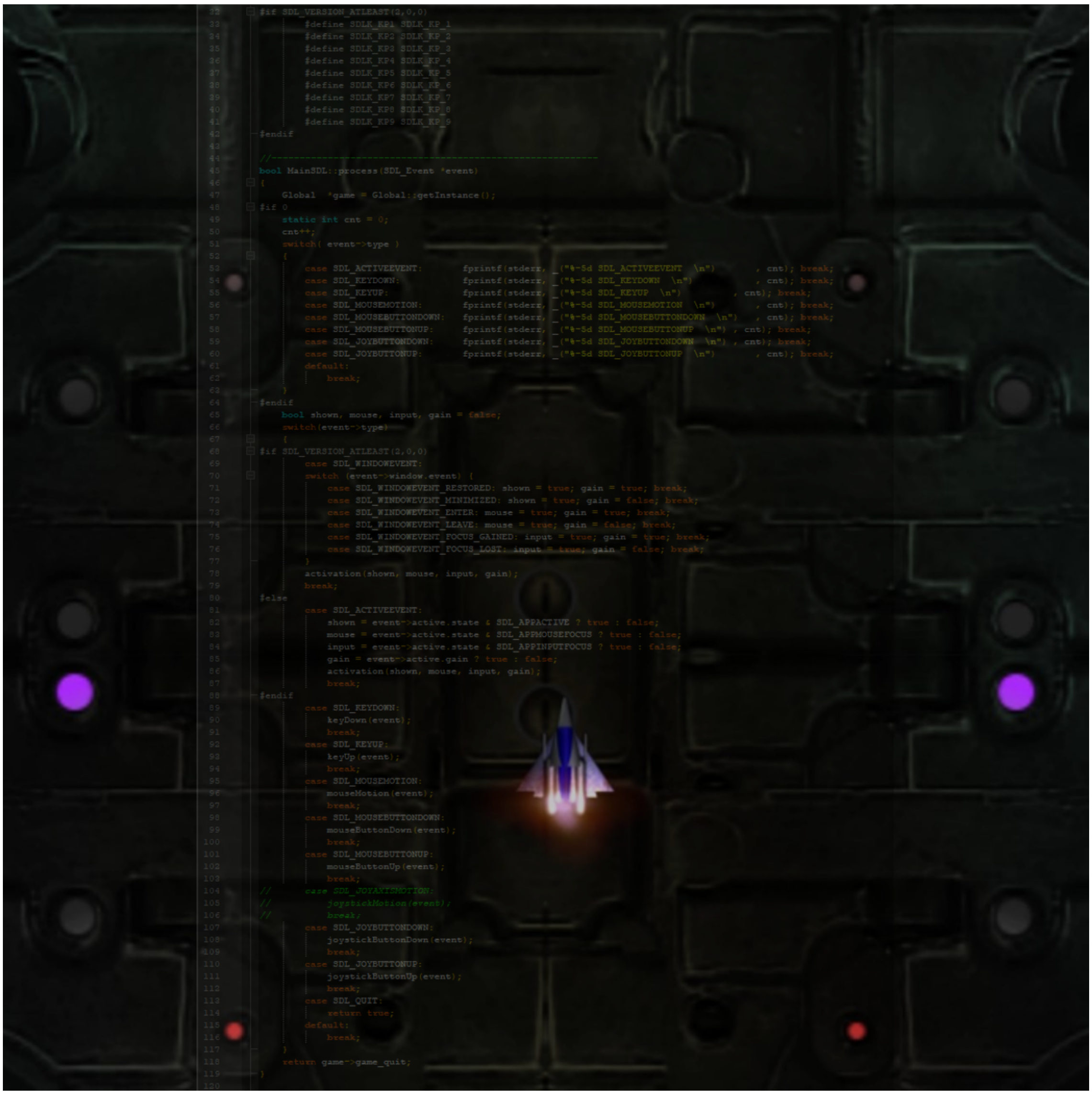
SHMU-C

Concept Document

Author:

Rishi Krishnan Sherly



```
#if SDL_VERSION_ATLEAST(2,0,0)
        #define SDLK_KP1 SDLK_KP_1
        #define SDLK_KP2 SDLK_KP_2
        #define SDLK_KP3 SDLK_KP_3
        #define SDLK_KP4 SDLK_KP_4
        #define SDLK_KP5 SDLK_KP_5
        #define SDLK_KP6 SDLK_KP_6
        #define SDLK_KP7 SDLK_KP_7
        #define SDLK_KP8 SDLK_KP_8
        #define SDLK_KP9 SDLK_KP_9
#endif

//------------------------------------------------------------
bool MainSDL::process(SDL_Event *event)
{
    Global  *game = Global::getInstance();
#if 0
    static int cnt = 0;
    cnt++;
    switch( event->type )
    {
        case SDL_ACTIVEEVENT:       fprintf(stderr, _("%-5d SDL_ACTIVEEVENT  \n")        , cnt); break;
        case SDL_KEYDOWN:           fprintf(stderr, _("%-5d SDL_KEYDOWN  \n")            , cnt); break;
        case SDL_KEYUP:             fprintf(stderr, _("%-5d SDL_KEYUP  \n")          , cnt); break;
        case SDL_MOUSEMOTION:       fprintf(stderr, _("%-5d SDL_MOUSEMOTION  \n")        , cnt); break;
        case SDL_MOUSEBUTTONDOWN:   fprintf(stderr, _("%-5d SDL_MOUSEBUTTONDOWN  \n")    , cnt); break;
        case SDL_MOUSEBUTTONUP:     fprintf(stderr, _("%-5d SDL_MOUSEBUTTONUP  \n") , cnt); break;
        case SDL_JOYBUTTONDOWN:     fprintf(stderr, _("%-5d SDL_JOYBUTTONDOWN  \n") , cnt); break;
        case SDL_JOYBUTTONUP:       fprintf(stderr, _("%-5d SDL_JOYBUTTONUP  \n")        , cnt); break;
        default:
            break;
    }
#endif
    bool shown, mouse, input, gain = false;
    switch(event->type)
    {
#if SDL_VERSION_ATLEAST(2,0,0)
        case SDL_WINDOWEVENT:
        switch (event->window.event) {
            case SDL_WINDOWEVENT_RESTORED: shown = true; gain = true; break;
            case SDL_WINDOWEVENT_MINIMIZED: shown = true; gain = false; break;
            case SDL_WINDOWEVENT_ENTER: mouse = true; gain = true; break;
            case SDL_WINDOWEVENT_LEAVE: mouse = true; gain = false; break;
            case SDL_WINDOWEVENT_FOCUS_GAINED: input = true; gain = true; break;
            case SDL_WINDOWEVENT_FOCUS_LOST: input = true; gain = false; break;
        }
        activation(shown, mouse, input, gain);
        break;
#else
        case SDL_ACTIVEEVENT:
            shown = event->active.state & SDL_APPACTIVE ? true : false;
            mouse = event->active.state & SDL_APPMOUSEFOCUS ? true : false;
            input = event->active.state & SDL_APPINPUTFOCUS ? true : false;
            gain = event->active.gain ? true : false;
            activation(shown, mouse, input, gain);
            break;
#endif
        case SDL_KEYDOWN:
            keyDown(event);
            break;
        case SDL_KEYUP:
            keyUp(event);
            break;
        case SDL_MOUSEMOTION:
            mouseMotion(event);
            break;
        case SDL_MOUSEBUTTONDOWN:
            mouseButtonDown(event);
            break;
        case SDL_MOUSEBUTTONUP:
            mouseButtonUp(event);
            break;
//      case SDL_JOYAXISMOTION:
//          joystickMotion(event);
//          break;
        case SDL_JOYBUTTONDOWN:
            joystickButtonDown(event);
            break;
        case SDL_JOYBUTTONUP:
            joystickButtonUp(event);
            break;
        case SDL_QUIT:
            return true;
        default:
            break;
    }
    return game->game_quit;
}
```

1 Overview

SHMU-C (Shoot 'em up - Code) is an educational top-scrolling 2D shoot 'em up, where the player controls a spaceship through the levels by writing simple code.

2 Gameplay

The game revolves around learning how to navigate the ship through static levels. The levels are static so as to allow the player to learn through mistakes and progress.

2.1 Ship

The ship moves up at a constant speed automatically and can move to either side (left or right) or shoot obstacles through coding. The ship also has a more powerful cannon, which is needed to take down certain obstacles, but has a slower reload time.

2.2 Obstacles

While navigating the levels, the player will come across various obstacles, such as asteroids, enemy ships and boss ships. Obstacles spawn from the top and move towards the bottom.

2.2.1 Asteroids

Asteroids appear from the top and move in a static path downwards (as the player flies past them). The player ship gets destroyed if hit by the asteroid and the asteroids cannot be destroyed by the normal cannon. However, the player can avoid it by flying around or use it as a shield against enemy fire.

2.2.2 Enemy Ships

Enemy ships also spawn at the top and move downwards in a static path. They shoot at specific intervals and can be destroyed with either of the player cannons. Being hit by either the enemy ship or their projectiles will destroy the player ship.

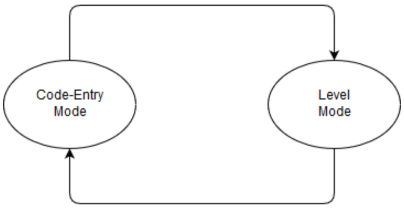
2.2.3 Boss Ships

The boss ships are the only obstacles with non-static behaviour. They move in a more unpredictable way and fire in random patterns. Their patterns are random and might change each play through. They also survive more damage before getting destroyed.

2.3 Game Modes

The game has two modes, the code entry mode and the level mode. In the code entry mode, the player enters code for controlling the ship. Once done, the player can start the level mode, in which the code is executed. The ship moves according to the code and gets destroyed if it collides with any of the obstacles. If the ship does get destroyed, the level ends and the game goes back into code entry mode. The line of code corresponding to the destruction of the ship gets highlighted as well, to make it easier to improve upon.

The core loop of the game is as follows.



3 Controls

The ship is controlled by writing simple pseudo code instructions. The following are possible through the code.

- Move : Moves the ship in the specified direction. Can also specify the start time and duration.
- Shoot : Shoots the cannon (or the special cannon) at the specified time.
- Macro : A set of instructions can be placed in a macro and combined into a single command that is run at a specified time.
- Conditionals : Allows more versatile control of the ship through conditions such as "Enemy in front of ship", "Being shot at", "Cannon Ready" and so on.

4 Aesthetics

The aesthetic goal of the game is to create a constant sense of progression, even when failing. This is done by the core loop of the game that switches between writing and testing the code. Each iteration of the core loop takes the player closer to completing the level.

4.1 Visuals

The visual style should be sci-fi or sci-fantasy.

4.2 Sounds

The music theme should be modern sci-fi. The pacing should match that of the level.

5 Target Audience

The game is designed to teach the basics of programming. As such, the target audience is those with an interest in learning programming. It is meant for those in the age group of 15 to 25, but can be played by anyone with an interest in learning programming or pre-planned strategies.