```cpp
#include "Game.h"
#include <SDL.h>
#include <stdio.h>
#include <SDL_image.h>
#include "Window.h"
#include "SDL_ttf.h"
#include "SDL_mixer.h"
#include "Mixer.h"

Game::Game(bool fullscreen) {
    runningFlag = false;
    restartFlag = false;

    //Setting up Window Singleton
    Window* window = Window::getInstance();

    if (!window->setFullscreen(fullscreen)) {
        printf("Unable to set Window::window->fullscreen to %d\n",
          fullscreen);
    }

    if (!window->setSize(640, 400)) {
        printf("Unable to set window size\n");
    }

    lvl_elements.clear();
    menu = NULL;
};

Game::Game(int width, int height) {
    runningFlag = false;
    restartFlag = false;

    //Setting up Singleton Window
    Window* window = Window::getInstance();

    if (!window->setFullscreen(false)) {
        printf("Unable to set Window::window->fullscreen to %d\n", false);
    }

    if (!window->setSize(640, 400)) {
        printf("Unable to set Window Size");
    }

    lvl_elements.clear();
    menu = NULL;
};

bool Game::init() {
    bool success = true;
    SDL_Window* global_window = NULL;
    SDL_Renderer* renderer = NULL;
    int window_size_w = Window::getInstance()->getWindowSizeW();
```

```
53      int window_size_h = Window::getInstance()->getWindowSizeH();
54
55      //Initializing SDL
56      if (SDL_Init(SDL_INIT_EVERYTHING) < 0) {
57          printf("Failed to initialize SDL! SDL_Error: %s\n", SDL_GetError
              ());
58          success = false;
59      }
60      else {
61
62          //Set texture filtering to linear
63          if (!SDL_SetHint(SDL_HINT_RENDER_SCALE_QUALITY, "1"))
64          {
65              printf("Warning: Linear texture filtering not enabled!");
66          }
67
68          //Creating the window
69          if (Window::getInstance()->isFullscreen()) {
70              global_window = SDL_CreateWindow("Prototype1",
                  SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED,
                  window_size_w, window_size_h, SDL_WINDOW_FULLSCREEN_DESKTOP);
71          }
72          else {
73              global_window = SDL_CreateWindow("Prototype1",
                  SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED,
                  window_size_w, window_size_h, SDL_WINDOW_SHOWN);
74          }
75          if (global_window == NULL) {
76              printf("Failed to create window! SDL_Error: %s\n",
                  SDL_GetError());
77              success = false;
78          }
79          else {
80
81              //Create renderer for window
82              renderer = SDL_CreateRenderer(global_window, -1,
                  SDL_RENDERER_ACCELERATED);
83              if (renderer == NULL) {
84                  printf("Renderer could not be created! SDL Error: %s\n",
                      SDL_GetError());
85                  success = false;
86              }
87              else {
88
89                  //Initialize renderer color
90                  SDL_SetRenderDrawColor(renderer, 0xFF, 0xFF, 0xFF, 0xFF);
91
92                  //Initialize PNG loading
93                  int imgFlags = IMG_INIT_PNG;
94                  if (!(IMG_Init(imgFlags) & imgFlags)) {
95                      printf("SDL_image could not initialize! SDL_image
                        Error: %s\n", IMG_GetError());
96                      success = false;
```

```
 97                  }
 98
 99                  //Initialize SDL_ttf
100                  if (TTF_Init() == -1)
101                  {
102                      printf("SDL_ttf could not initialize. SDL_ttf Error: %
                     s\n", TTF_GetError());
103                      success = false;
104                  }
105
106                  //Initialize SDL_mixer
107                  if( Mix_OpenAudio( 44100, MIX_DEFAULT_FORMAT, 2, 2048 ) <
                       0 ) {
108                      printf( "SDL_mixer could not initialize. SDL_mixer
                     Error: %s\n", Mix_GetError() );
109                      success = false;
110                  }
111
112                  //Disable Mouse
113                  SDL_ShowCursor(SDL_DISABLE);
114
115                  //adapt global window size variables
116                  if (Window::getInstance()->isFullscreen()) {
117                      SDL_GetWindowSize(global_window, &window_size_w,
                     &window_size_h);
118                      if (!Window::getInstance()->setSize(window_size_w,
                     window_size_h)) {
119                          printf("Unable to adapt window size variables");
120                      }
121                  }
122
123                  if (!Window::getInstance()->setWindowAndRenderer
                       (global_window, renderer)) {
124                      printf("Window Could not be set up. Failure in
                     Window::setWindow(..).");
125                      success = false;
126                  }
127
128                  //Create Menu
129                  menu = new Menu();
130                  if (menu->init() != true) {
131                      printf("Failed to initiate Menu.\n");
132                  }
133
134                  //Setting up the Player
135                  PlayerLeft* pl = new PlayerLeft();
136                  PlayerRight* pr = new PlayerRight();
137                  lvl_elements.push_back(pl);
138                  lvl_elements.push_back(pr);
139
140                  //Attach them to the Collision Detection System. If one
                       attachment fails print error
141                  ObservableCollisionDetection* cd =
```

```
                    ObservableCollisionDetection::getInstance();
142                 if ((!cd->attach(pl)) || (!cd->attach(pr))) {
143                     printf("Failed to attach one or both players to
                       Collision Detection System");
144                 }
145                 pl = NULL;
146                 pr = NULL;
147
148                 //Attach Background
149                 lvl_elements.push_back(new Background());
150             }
151         }
152     }
153
154     global_window = NULL;
155     renderer = NULL;
156
157     return success;
158 }
159
160 bool Game::load_media() {
161     bool success = true;
162
163     //Load lvl elements
164     for (int i = 0; i < lvl_elements.size(); i++) {
165         if (!lvl_elements.at(i)->loadMedia()) {
166             printf("Failed to load lvl element: %s! SDL Error: %s\n",
                  lvl_elements.at(i)-> getType().c_str(), SDL_GetError());
167             success = false;
168         }
169     }
170
171     if (!menu->loadMedia()) {
172         printf("Failed to load Menu! SDL Error: %s\n", SDL_GetError());
173         success = false;
174     }
175
176     if (!Mixer::getInstance()->loadMedia()) {
177         printf("Failed to load sound effects. SDL_mixer Error: %s\n",
              Mix_GetError());
178         success = false;
179     }
180
181     return success;
182 };
183
184 void Game::start() {
185     printf("Loading...");
186     if (init() == true) {
187         if (load_media() == true) {
188             printf("Done\n");
189             if (menu->show(Menu::STARTGAME) == true) {
190                 run();
```

```
191             }
192         }
193         else {
194             printf("Failed to load Media!\n");
195         }
196     }
197     else {
198         printf("Failed to initialize!\n");
199     }
200 };
201
202 void Game::run() {
203     //The Timing
204     const int FPS = 60;
205     const int frame_delay = 1000 / FPS; // How much ms must pass for each
        frame
206     Uint32 frame_start = 0;
207     int frame_time = 0;
208
209     //set flags
210     runningFlag = true;
211     restartFlag = false;
212
213     //Game loop
214     while (runningFlag) {
215         frame_start = SDL_GetTicks();
216
217         input();
218         update();
219         render();
220
221         //Check if restartFlag is true
222         if (restartFlag == true) {
223             restart();
224             restartFlag = false;
225             runningFlag = true;
226         }
227
228         //If Frame is processed before its appropriate time
229         frame_time = SDL_GetTicks() - frame_start;
230         if (frame_delay > frame_time) {
231             frame_time = frame_delay - frame_time;
232             SDL_Delay(frame_time);
233         }
234     }
235 }
236
237 void Game::input() {
238     //Check ESC and Quit Button
239     SDL_Event e;
240     while (SDL_PollEvent(&e)) {
241
242         if (e.type == SDL_QUIT) {
```

```
243             runningFlag = false;
244         }
245
246         if (e.type == SDL_KEYDOWN) {
247             if (e.key.keysym.sym == SDLK_ESCAPE) {
248                 Mixer::getInstance()->playMusic(Mixer::LOBBYSONG);
249                 if (gameEnded() == true) {
250                     if (menu->show(Menu::RESTART) == false) {
251                         runningFlag = false;
252                     }
253                     else {
254                         restartFlag = true;
255                     }
256                 }
257                 else {
258                     if (menu->show(Menu::CONTINUE) == false) {
259                         runningFlag = false;
260                     }
261                 }
262             }
263         }
264     }
265
266     //check player inputs
267     for (int i = 0; i < lvl_elements.size(); i++) {
268         lvl_elements.at(i)->checkInput();
269     }
270
271     //Check if a bottle is thrown if yes try to spawn a bottle but check
        NULL pointer parameter
272     const Uint8* currentKeyStates = SDL_GetKeyboardState(NULL);
273
274     if (currentKeyStates[SDL_SCANCODE_I]) {
275         for (int i = 0; i < lvl_elements.size(); i++) {
276             if (lvl_elements.at(i)->getType().compare("PLAYERRIGHT") == 0)
                {
277                 Bottle* bottle = lvl_elements.at(i)->spawnBottle();
278                 if (bottle != NULL){
279                     lvl_elements.push_back(bottle);
280                 }
281                 break;
282             }
283         }
284     }
285
286     if (currentKeyStates[SDL_SCANCODE_C]) {
287         for (int i = 0; i < lvl_elements.size(); i++) {
288             if (lvl_elements.at(i)->getType().compare("PLAYERLEFT") == 0)
                {
289                 Bottle* bottle = lvl_elements.at(i)->spawnBottle();
290                 if (bottle != NULL) {
291                     lvl_elements.push_back(bottle);
292                 }
```

```
293                 break;
294             }
295         }
296     }
297 };
298 
299 void Game::tick() {
300     for (int i = 0; i < lvl_elements.size(); i++) {
301         lvl_elements.at(i)->tick();
302     }
303 };
304 
305 void Game::restart() {
306     for (int i = 0; i < lvl_elements.size(); i++) {
307         lvl_elements.at(i)->restart();
308     }
309 };
310 
311 void Game::update() {
312     //Check Collisions
313     ObservableCollisionDetection::getInstance()->checkCollisions();
314 
315     //CHECK IF BOTTLES ARE BROKEN AND READY TO GET DETACHED FROM GAME
316     for (int i = 0; i < lvl_elements.size(); i++) {
317         if (lvl_elements.at(i)->isDead().compare("BROKENBOTTLE") == 0) {
318             LevelElementInterface* tmp = lvl_elements.at(i);
319             lvl_elements.erase(lvl_elements.begin() + i);
320         }
321     }
322 
323     //tick players
324     tick();
325 };
326 
327 void Game::render() {
328     SDL_Renderer* renderer = Window::getInstance()->getRenderer();
329 
330     //Clear Screen
331     SDL_RenderClear(renderer);
332 
333     //Render Background
334     int bg = 0;
335     for (int i = 0; i < lvl_elements.size(); i++) {
336         if (lvl_elements.at(i)->getType().compare("BACKGROUND") == 0) {
337             bg = i;
338             lvl_elements.at(i)->render();
339         }
340     }
341 
342     //Render other lvl elements
343     for (int i = 0; i < lvl_elements.size(); i++) {
344         if(i != bg)
345             lvl_elements.at(i)->render();
```

```
346     }
347
348     //Update screen
349     SDL_RenderPresent(renderer);
350 };
351
352 bool Game::gameEnded() {
353     bool someoneDied = false;
354
355     for (int i = 0; i < lvl_elements.size(); i++) {
356         if (lvl_elements.at(i)->isDead().compare("DEADPLAYER") == 0) {
357             someoneDied = true;
358             break;
359         }
360     }
361
362     return someoneDied;
363 };
364
365 void Game::close() {
366     //Destroy Players
367     for (int i = 0; i < lvl_elements.size(); i++) {
368         lvl_elements.at(i)->close();
369     }
370     lvl_elements.clear();
371
372     //Destroy Window, Collision Detection and Mixer - Singletons
373     Window::getInstance()->~Window();
374     ObservableCollisionDetection::getInstance()->~ObservableCollisionDetection();
375     Mixer::getInstance()->~Mixer();
376
377     //Close Menu
378     menu->close();
379
380     //Quit SDL Subsystems
381     Mix_Quit();
382     TTF_Quit();
383     IMG_Quit();
384     SDL_Quit();
385 };
386
387 Game::~Game() {
388     close();
389 };
```