```cpp
#include "SDL.h"
#include "Background.h"
#include "SDL_image.h"
#include "Window.h"

Background::Background() {
    for (int i = 0; i < far_buildings.size(); i++) {
        far_buildings[i] = NULL;
    }
    for (int i = 0; i < foregrounds.size(); i++) {
        foregrounds[i] = NULL;
    }
    filling_up_the_bottle = true;
    backgroundTX = NULL;
    far_buildings_ticker = 0;
    foreground_ticker = 0;
    for (int i = 0; i < bats.size(); i++) {
        bats[i] = NULL;
    }
};

bool Background::loadMedia() {
    bool success = true;

    //Set up bats
    for (int i = 0; i < bats.size(); i++) {
        bats[i] = new Bat();
    }

    //FAR BUILDINGS
    //set up an array with the paths for all the far buildings
    std::array<std::string, 5> paths;
    for (int i = 0; i < paths.size(); i++) {
        paths[i] = "assets/sprite_sheets/background/far_buildings/
          far_buildings" + std::to_string(i + 1) + ".png";
    }

    //Load far buildings
    for (int i = 0; i < far_buildings.size()-1; i++) {
        far_buildings[i+1] = loadTexture(paths[i].c_str());
        if (far_buildings[i+1] == NULL) {
            printf("Failed to create texture, far_buildings[%d]. SDL
              Error: %s\n",i+1, SDL_GetError());
            success = false;
        }
    }
    far_buildings[0] = far_buildings[1];

    //BACKGROUND BUILDINGS
    //Load background buildings
    backgroundTX = loadTexture("assets/sprite_sheets/background/
      background_buildings.png");
    if (backgroundTX == NULL) {
```

```
51          printf("Failed to create texture. SDL Error: %s\n", SDL_GetError
              ());
52          success = false;
53      }
54
55      //FOREGROUND BUILDINGS
56      //set up an array with the paths for all the far buildings
57      std::array<std::string, 25> paths_foreground;
58      for (int i = 0; i < paths_foreground.size(); i++) {
59          paths_foreground[i] = "assets/sprite_sheets/background/foreground/
              foreground" + std::to_string(i + 1) + ".png";
60      }
61
62      //Load foregrounds
63      for (int i = 0; i < foregrounds.size() - 1; i++) {
64          foregrounds[i + 1] = loadTexture(paths_foreground[i].c_str());
65          if (foregrounds[i + 1] == NULL) {
66              printf("Failed to create texture, foregrounds[%d]. SDL Error:
                    %s\n", i + 1, SDL_GetError());
67              success = false;
68          }
69      }
70      foregrounds[0] = foregrounds[25];
71
72      //BATS
73      for (int i = 0; i < bats.size(); i++) {
74          //Load Bats
75          if (!bats[i]->loadMedia()) {
76              printf("Failed to create texture. SDL Error: %s\n",
                    SDL_GetError());
77              success = false;
78          }
79      }
80
81      return success;
82  };
83
84  void Background::tick() {
85      //FAR BUILDINGS
86      //Tick and Choose Far Building Texture
87      far_buildings_ticker++;
88
89      if (far_buildings_ticker > 120) {
90          far_buildings[0] = far_buildings[rand() % 5 + 1];
91          far_buildings_ticker = 0;
92      }
93      if (far_buildings_ticker > 20) {
94          far_buildings[0] = far_buildings[1];
95      }
96
97      //FOREGROUND
98      foreground_ticker++;
99      int number_of_ticks_till_tx_change = 60;
```

```
100
101     //Filling up the bottle animation
102     if ((foregrounds[0] == foregrounds[25]) && (foreground_ticker >
          number_of_ticks_till_tx_change)) {
103         filling_up_the_bottle = true;
104         number_of_ticks_till_tx_change = 2;
105     }
106     //To ensure that the texture gets changed after two ticks, which is
          only
107     //the case when we're filling up the bottle
108     if (filling_up_the_bottle == true) {
109         number_of_ticks_till_tx_change = 2;
110     }
111
112     //Changing the current image to the next one
113     if ((foreground_ticker > number_of_ticks_till_tx_change) &&
          (filling_up_the_bottle == false)) {
114         foreground_ticker = 0;
115         //finding out what spritesheet we're currently at and changing it
            to the next one
116         for (int i = 0; i < foregrounds.size(); i++) {
117             if ((foregrounds[0] == foregrounds[i]) && (i!=0)) {
118                 if (i == foregrounds.size() - 1) {
119                     foregrounds[0] = foregrounds[1];
120                 }
121                 else {
122                     foregrounds[0] = foregrounds[i + 1];
123                     break;
124                 }
125             }
126         }
127     }
128     //Filling up bottle
129     else if ((foreground_ticker > number_of_ticks_till_tx_change) &&
          (filling_up_the_bottle == true)) {
130         foreground_ticker = 0;
131         //finding out what spritesheet we're currently at
132         for (int i = foregrounds.size()-1; i > 0; i--) {
133             if ((foregrounds[0] == foregrounds[i]) && (i != 0)) {
134                 //changing it to the next one
135                 if (i == 1) {
136                     filling_up_the_bottle = false;
137                     number_of_ticks_till_tx_change = 150;
138                 }
139                 else {
140                     foregrounds[0] = foregrounds[i - 1];
141                     break;
142                 }
143             }
144         }
145     }
146
147     //Tick Bats
```

```
148     for (int i = 0; i < bats.size(); i++) {
149         bats[i]->tick();
150     }
151 };
152 
153 void Background::render() {
154     //Get Renderer
155     SDL_Renderer* renderer = Window::getInstance()->getRenderer();
156 
157     //Save old viewport
158     SDL_Rect old_viewport;
159     SDL_RenderGetViewport(renderer, &old_viewport);
160 
161     //Create viewport
162     SDL_Rect topLeftViewport;
163     topLeftViewport.x = 0;
164     topLeftViewport.y = 0;
165     topLeftViewport.w = Window::getInstance()->getWindowSizeW();
166     topLeftViewport.h = Window::getInstance()->getWindowSizeH();
167 
168     //set viewport
169     SDL_RenderSetViewport(renderer, &topLeftViewport);
170 
171     //Render far_buldings to screen
172     SDL_RenderCopy(renderer, far_buildings[0], NULL, NULL);
173 
174     //Render back buildings to screen
175     SDL_RenderCopy(renderer, backgroundTX, NULL, NULL);
176 
177     //Bats Flying in the back of the buildings
178     for (int i = 0; i < bats.size(); i++) {
179         if(bats[i]->getZIndex() == 1)
180             bats[i]->render();
181     }
182 
183     //Render foreground to frame
184     SDL_RenderCopy(renderer, foregrounds[0], NULL, NULL);
185 
186     //Bats Flying in front of the building
187     for (int i = 0; i < bats.size(); i++) {
188         if (bats[i]->getZIndex() == 2)
189             bats[i]->render();
190     }
191 
192     //Reset viewport
193     SDL_RenderSetViewport(renderer, &old_viewport);
194 }
195 
196 void Background::restart() {
197     for (int i = 0; i < bats.size(); i++) {
198         bats[i]->~Bat();
199         bats[i] = NULL;
200     }
```

```
201
202     for (int i = 0; i < bats.size(); i++) {
203         bats[i] = new Bat();
204         bats[i]->loadMedia();
205     }
206
207     filling_up_the_bottle = true;
208     far_buildings_ticker = 0;
209     foreground_ticker = 0;
210     far_buildings[0] = far_buildings[1];
211     foregrounds[0] = foregrounds[25];
212 };
213
214 void Background::close() {
215     //Destroy Bats
216     for (int i = 0; i < bats.size(); i++) {
217         bats[i]->~Bat();
218         bats[i] = NULL;
219     }
220
221     //Destroy Far Buildings
222     for (int i = 0; i < far_buildings.size(); i++) {
223         SDL_DestroyTexture(far_buildings[i]);
224         far_buildings[i] = NULL;
225     }
226
227     //Destroy Background tx
228     SDL_DestroyTexture(backgroundTX);
229     backgroundTX = NULL;
230
231     //Destroy Foregrounds
232     for (int i = 0; i < foregrounds.size(); i++) {
233         SDL_DestroyTexture(foregrounds[i]);
234         foregrounds[i] = NULL;
235     }
236 }
237
238 std::string Background::getType() { return "BACKGROUND"; };
239
240 Bottle* Background::spawnBottle() { return NULL; };
241
242 void Background::checkInput() { return; };
243
244 std::string Background::isDead() { return "NOTDEAD"; };
245
246 Background::~Background() {
247     close();
248 };
```