

```
1 #include "PlayerLeft.h"
2 #include "Window.h"
3 #include "ObservableCollisionDetection.h"
4 #include "Mixer.h"
5
6 PlayerLeft::PlayerLeft() : Player(false){
7     appearance.x = 50;
8     headLeft = false;
9 };
10
11 void PlayerLeft::checkInput() {
12     //CONDITION: If dead don't take anymore inputs
13     if (healthbar->isEmpty() == true) {
14         return;
15     }
16
17     //CHECKING KEYBOARD INPUTS
18     const Uint8* currentKeyStates = SDL_GetKeyboardState(NULL);
19
20     //S Button Pressed
21     if (currentKeyStates[SDL_SCANCODE_S]) {
22         if (curr_state[1] != BLOCK) { //{current picture of state/
23             animation, current state}
24             changeStateTo(BLOCK);
25             blocking = true;
26             Mixer::getInstance()->play(Mixer::BLOCK);
27         }
28         return;
29     }
30     else { blocking = false; }
31
32     //D Button Pressed
33     if (currentKeyStates[SDL_SCANCODE_D]) {
34         headLeft = false;
35         moveX(false, 4);
36         if (curr_state[1] != WALK) {
37             changeStateTo(WALK);
38         }
39     }
40
41     //A Button Pressed
42     if (currentKeyStates[SDL_SCANCODE_A]) {
43         headLeft = true;
44         moveX(false, 4);
45         if (curr_state[1] != WALK) {
46             changeStateTo(WALK);
47         }
48     }
49
50     //W Button Pressed
51     if (currentKeyStates[SDL_SCANCODE_W]) {
52         if (curr_state[1] != JUMP) {
53             Mixer::getInstance()->play(Mixer::JUMP);
```

```
53         changeStateTo(JUMP);
54     }
55 }
56
57 //V Button Pressed
58 if (currentKeyStates[SDL_SCANCODE_V]) {
59     if (curr_state[1] != STAB) {
60         changeStateTo(STAB);
61         Mixer::getInstance()->play(Mixer::SWING);
62     }
63 }
64
65 //C Button Pressed
66 if (currentKeyStates[SDL_SCANCODE_C]) {
67     if (curr_state[1] != THROWBOTTLE) {
68         changeStateTo(THROWBOTTLE);
69         Mixer::getInstance()->play(Mixer::SWING);
70     }
71 }
72 };
73
74 void PlayerLeft::tick() {
75     //If dead don't do anything
76     if (healthbar->isEmpty() == true) {
77         changeStateTo(BLOCK);
78         blocking = true;
79     }
80
81     float increaseFactor = 15;
82     //JUMPING
83     //Increase height if jumped
84     if ((curr_state[1] == JUMP) && (heightAboveTheGround < 23)) {
85         float PI = 3.14159265;
86         heightAboveTheGround += 1;
87         float sinus = sin((float)heightAboveTheGround*(0.5*PI) / 23);
88         sinus = ((1 - sinus) * increaseFactor);
89         height_stack.push((int)sinus);
90         appearance.y -= (int)sinus;
91     }
92
93     //Decrease height after jump
94     if ((curr_state[1] != JUMP) && (heightAboveTheGround > 0)) {
95         heightAboveTheGround -= 1;
96         if (!height_stack.empty()) {
97             int tmp = height_stack.top();
98             height_stack.pop();
99             appearance.y += tmp;
100         }
101     }
102
103     //IF HURT
104     if (curr_state[1] == HURT) {
105         moveX(true, 15);
```

```
106     }
107
108     //Ticking the Player
109     ++ticked;
110     if (ticked >= 5) {
111         //Ensuring that the sprite isn't changed after the first animation ↗
112         //when blocking
113         if ((curr_state[1] == BLOCK) && (curr_state[0] == 7) && (blocking ↗
114             == true)) { ticked = 0; }
115         else {
116             ++curr_state[0];
117             //If the current_state is higher than there are sprites ↗
118             //available -> IDLE
119             if (curr_state[0] >= max_sprites[curr_state[1]]) {
120                 curr_state[0] = 0;
121                 curr_state[1] = IDLE;
122             }
123             ticked = 0;
124         }
125     }
126 };
127
128 void PlayerLeft::restart() {
129     int window_size_h = Window::getInstance()->getWindowSizeH();
130     int window_size_w = Window::getInstance()->getWindowSizeW();
131
132     //Set size and position
133     appearance.x = 50;
134     appearance.y = window_size_h - (window_size_h / 4);
135     appearance.w = 128;
136     appearance.h = 128;
137
138     //set direction and other flags
139     headLeft = false;
140     blocking = false;
141     ticked = 0;
142
143     //Set state to start with
144     curr_state[0] = 0;
145     curr_state[1] = IDLE;
146
147     //Set Height
148     heightAboveTheGround = 0;
149     while (!height_stack.empty()) {
150         height_stack.pop();
151     }
152
153     //Refill Healthbar
154     healthbar->refill();
155 };
156
157 std::string PlayerLeft::getType() {
158     return "PLAYERLEFT";
159 }
```

```
156 };
```

```
157
```

```
158 PlayerLeft::~~PlayerLeft() {};
```