```
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;
       headLeft = false;
8
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
       if (currentKeyStates[SDL_SCANCODE_S]) {
21
           if (curr_state[1] != BLOCK) { //{current picture of state/
22
              animation, current state}
23
                changeStateTo(BLOCK);
24
                blocking = true;
25
                Mixer::getInstance()->play(Mixer::BLOCK);
26
           }
27
           return;
28
29
       else { blocking = false; }
30
       //D Button Pressed
31
       if (currentKeyStates[SDL_SCANCODE_D]) {
32
           headLeft = false;
33
34
           moveX(false, 4);
           if (curr_state[1] != WALK) {
35
36
                changeStateTo(WALK);
37
           }
38
       }
39
40
       //A Button Pressed
41
       if (currentKeyStates[SDL_SCANCODE_A]) {
           headLeft = true;
42
           moveX(false, 4);
43
           if (curr_state[1] != WALK) {
44
45
                changeStateTo(WALK);
46
           }
47
       }
48
49
       //W Button Pressed
50
       if (currentKeyStates[SDL SCANCODE W]) {
           if (curr_state[1] != JUMP) {
51
52
                Mixer::getInstance()->play(Mixer::JUMP);
```

```
53
                 changeStateTo(JUMP);
 54
             }
 55
         }
56
 57
         //V Button Pressed
         if (currentKeyStates[SDL_SCANCODE_V]) {
 58
 59
             if (curr state[1] != STAB) {
                 changeStateTo(STAB);
 60
                 Mixer::getInstance()->play(Mixer::SWING);
 61
 62
             }
 63
        }
 64
 65
         //C Button Pressed
         if (currentKeyStates[SDL SCANCODE C]) {
 66
 67
             if (curr_state[1] != THROWBOTTLE) {
 68
                 changeStateTo(THROWBOTTLE);
                 Mixer::getInstance()->play(Mixer::SWING);
 69
 70
             }
         }
 71
 72 };
 73
    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
        if ((curr_state[1] == JUMP) && (heightAboveTheGround < 23)) {</pre>
 84
             float PI = 3.14159265;
 85
             heightAboveTheGround += 1;
 86
 87
             float sinus = sin((float)heightAboveTheGround*(0.5*PI) / 23);
             sinus = ((1 - sinus) * increaseFactor);
 88
 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;
             if (!height_stack.empty()) {
 96
 97
                 int tmp = height_stack.top();
 98
                 height_stack.pop();
 99
                 appearance.y += tmp;
100
             }
         }
101
102
         //IF HURT
103
         if (curr_state[1] == HURT) {
104
105
             moveX(true, 15);
```

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

155

return "PLAYERLEFT";

```
C:\Dev\FinalProject\ClassFiles\PlayerLeft.cpp
156 };
```

```
4
```

```
156 };
157
158 PlayerLeft::~PlayerLeft() {};
```