```cpp
1  #include "PlayerRight.h"
2  #include "Window.h"
3  #include "ObservableCollisionDetection.h"
4  #include "Mixer.h"
5
6  PlayerRight::PlayerRight() : Player(true){
7      headLeft = true;
8      appearance.x = Window::getInstance()->getWindowSizeW() - 178;
9  };
10
11 void PlayerRight::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     //Down Button Pressed
21     if (currentKeyStates[SDL_SCANCODE_DOWN]) {
22         if (curr_state[1] != BLOCK) {
23             changeStateTo(BLOCK);
24             blocking = true;
25             Mixer::getInstance()->play(Mixer::BLOCK);
26         }
27         return;
28     }
29     else { blocking = false; }
30
31     //Right Button Pressed
32     if (currentKeyStates[SDL_SCANCODE_RIGHT]) {
33         headLeft = false;
34         moveX(false, 4);
35         if (curr_state[1] != WALK) {
36             changeStateTo(WALK);
37         }
38     }
39
40     //Left Button Pressed
41     if (currentKeyStates[SDL_SCANCODE_LEFT]) {
42         headLeft = true;
43         moveX(false, 4);
44         if (curr_state[1] != WALK) {
45             changeStateTo(WALK);
46         }
47     }
48
49     //Up Button Pressed
50     if (currentKeyStates[SDL_SCANCODE_UP]) {
51         if (curr_state[1] != JUMP) {
52             Mixer::getInstance()->play(Mixer::JUMP);
53             changeStateTo(JUMP);
```

```cpp
 54            }
 55        }
 56
 57        //O Button Pressed
 58        if (currentKeyStates[SDL_SCANCODE_O]) {
 59            if (curr_state[1] != STAB) {
 60                changeStateTo(STAB);
 61                Mixer::getInstance()->play(Mixer::SWING);
 62            }
 63        }
 64
 65        //I Button Pressed
 66        if (currentKeyStates[SDL_SCANCODE_I]) {
 67            //Make sure that the sound is only played once
 68            if (curr_state[1] != THROWBOTTLE) {
 69                changeStateTo(THROWBOTTLE);
 70                Mixer::getInstance()->play(Mixer::SWING);
 71            }
 72        }
 73 };
 74
 75 void PlayerRight::tick() {
 76        //If dead don't do anything
 77        if (healthbar->isEmpty() == true) {
 78            changeStateTo(BLOCK);
 79            blocking = true;
 80        }
 81
 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) * 10) + 3;
 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        }
```

```cpp
107
108        //Ticking the Player
109        ++ticked;
110        if (ticked >= 5) {
111            //Making sure that the sprite isn't changed after the first
                   animation when blocking
112            if ((curr_state[1] == BLOCK) && (curr_state[0] == 7) && (blocking
                   == true)) { ticked = 0; }
113            else {
114                ++curr_state[0];
115                //If the current_state is higher than there are sprites
                       available -> IDLE
116                if (curr_state[0] >= max_sprites[curr_state[1]]) {
117                    curr_state[0] = 0;
118                    curr_state[1] = IDLE;
119                }
120                ticked = 0;
121            }
122        }
123    };
124
125    void PlayerRight::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 = Window::getInstance()->getWindowSizeW() - 178;
131        appearance.y = window_size_h - (window_size_h / 4);
132        appearance.w = 128;
133        appearance.h = 128;
134
135        //Set direction and other flags
136        headLeft = true;
137        blocking = false;
138        ticked = 0;
139
140        //Set state to start with
141        curr_state[0] = 0;
142        curr_state[1] = IDLE;
143
144        //Set Height
145        heightAboveTheGround = 0;
146        while (!height_stack.empty()) {
147            height_stack.pop();
148        }
149
150        //Refill Healthbar
151        healthbar->refill();
152    };
153
154    std::string PlayerRight::getType() {
155        return "PLAYERRIGHT";
156    };
```

```
157
158  PlayerRight::~PlayerRight() {};
```