```cpp
#include "PlayerLeft.h"
#include "Window.h"
#include "ObservableCollisionDetection.h"
#include "Mixer.h"

PlayerLeft::PlayerLeft() : Player(false){
    appearance.x = 50;
    headLeft = false;
};

void PlayerLeft::checkInput() {
//CONDITION: If dead don't take anymore inputs
    if (healthbar->isEmpty() == true) {
        return;
    }

//CHECKING KEYBOARD INPUTS
    const Uint8* currentKeyStates = SDL_GetKeyboardState(NULL);

    //S Button Pressed
    if (currentKeyStates[SDL_SCANCODE_S]) {
        if (curr_state[1] != BLOCK) { //{current picture of state/
          animation, current state}
            changeStateTo(BLOCK);
            blocking = true;
            Mixer::getInstance()->play(Mixer::BLOCK);
        }
        return;
    }
    else { blocking = false; }

    //D Button Pressed
    if (currentKeyStates[SDL_SCANCODE_D]) {
        headLeft = false;
        moveX(false, 4);
        if (curr_state[1] != WALK) {
            changeStateTo(WALK);
        }
    }

    //A Button Pressed
    if (currentKeyStates[SDL_SCANCODE_A]) {
        headLeft = true;
        moveX(false, 4);
        if (curr_state[1] != WALK) {
            changeStateTo(WALK);
        }
    }

    //W Button Pressed
    if (currentKeyStates[SDL_SCANCODE_W]) {
        if (curr_state[1] != JUMP) {
            Mixer::getInstance()->play(Mixer::JUMP);
```

```
                changeStateTo(JUMP);
            }
        }

        //V Button Pressed
        if (currentKeyStates[SDL_SCANCODE_V]) {
            if (curr_state[1] != STAB) {
                changeStateTo(STAB);
                Mixer::getInstance()->play(Mixer::SWING);
            }
        }

        //C Button Pressed
        if (currentKeyStates[SDL_SCANCODE_C]) {
            if (curr_state[1] != THROWBOTTLE) {
                changeStateTo(THROWBOTTLE);
                Mixer::getInstance()->play(Mixer::SWING);
            }
        }
};

void PlayerLeft::tick() {
    //If dead don't do anything
    if (healthbar->isEmpty() == true) {
        changeStateTo(BLOCK);
        blocking = true;
    }

    float increaseFactor = 15;
    //JUMPING
    //Increase height if jumped
    if ((curr_state[1] == JUMP) && (heightAboveTheGround < 23)) {
        float PI = 3.14159265;
        heightAboveTheGround += 1;
        float sinus = sin((float)heightAboveTheGround*(0.5*PI) / 23);
        sinus = ((1 - sinus) * increaseFactor);
        height_stack.push((int)sinus);
        appearance.y -= (int)sinus;
    }

    //Decrease height after jump
    if ((curr_state[1] != JUMP) && (heightAboveTheGround > 0)) {
        heightAboveTheGround -= 1;
        if (!height_stack.empty()) {
            int tmp = height_stack.top();
            height_stack.pop();
            appearance.y += tmp;
        }
    }

    //IF HURT
    if (curr_state[1] == HURT) {
        moveX(true, 15);
```

```cpp
    }

    //Ticking the Player
    ++ticked;
    if (ticked >= 5) {
        //Ensuring that the sprite isn't changed after the first animation when blocking
        if ((curr_state[1] == BLOCK) && (curr_state[0] == 7) && (blocking == true)) { ticked = 0; }
        else {
            ++curr_state[0];
            //If the current_state is higher than there are sprites available -> IDLE
            if (curr_state[0] >= max_sprites[curr_state[1]]) {
                curr_state[0] = 0;
                curr_state[1] = IDLE;
            }
            ticked = 0;
        }
    }
};

void PlayerLeft::restart() {
    int window_size_h = Window::getInstance()->getWindowSizeH();
    int window_size_w = Window::getInstance()->getWindowSizeW();

    //Set size and position
    appearance.x = 50;
    appearance.y = window_size_h - (window_size_h / 4);
    appearance.w = 128;
    appearance.h = 128;

    //set direction and other flags
    headLeft = false;
    blocking = false;
    ticked = 0;

    //Set state to start with
    curr_state[0] = 0;
    curr_state[1] = IDLE;

    //Set Height
    heightAboveTheGround = 0;
    while (!height_stack.empty()) {
        height_stack.pop();
    }

    //Refill Healthbar
    healthbar->refill();
};

std::string PlayerLeft::getType() {
    return "PLAYERLEFT";
```

```
};

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