toaster.cpp

```
#include "toaster.hpp"
#include "tools.hpp"
#include "object.hpp"


Toaster::Toaster() : first_toast_position(870, -10), goal_y(-10)
{
    toast_tex.loadFromFile("sprites/toast.png");
    toast_sprite.setTexture(toast_tex);

    notch_tex.loadFromFile("sprites/LB_notch.png");
    notch_sprite.setTexture(notch_tex);
    notch_sprite.setPosition(leaderboard_position);
    notch_sprite.setScale(leaderboard_scale, leaderboard_scale);
    notch_sprite.setColor(sf::Color(255, 255, 255, 190));

    board_tex.loadFromFile("sprites/LB_player.png");
    board_sprite.setTexture(board_tex);
    board_sprite.setScale(leaderboard_scale, leaderboard_scale);
    board_sprite.setColor(sf::Color(255, 255, 255, 190));

    font.loadFromFile("Fonts/Roboto-Medium.ttf");

    text = sf::Text("Missing Text", font, 18);
    text.setFillColor(sf::Color(35, 35, 35));


    leaderboard_text = sf::Text("Missing Text", font, 18);
    leaderboard_text.setFillColor(sf::Color(35, 35, 35));
}


Toaster::LeaderboardEntry::LeaderboardEntry(int player_id, int score,
const string& username) :
    player_id(player_id), username(username), score(score),
    position_y(-100)
{

}

void Toaster::drawLeaderboard(sf::RenderWindow& window,
vector<LeaderboardEntry>& leaderboard, float dt)
{
    // empty leaderboard
    if (leaderboard.size() < 2) return;


    window.draw(notch_sprite);

    float initial_y = leaderboard_position.y + 14;
    for (int i = 0; i < leaderboard.size(); i++)
    {
        float desired_pos_y = initial_y + 34 * i;
```

```
        leaderboard[i].position_y = lerp(leaderboard[i].position_y,
        desired_pos_y, 0.1f);

    }

    for (int i = 0; i < leaderboard.size(); i++)
    {
        board_sprite.setPosition(leaderboard_position.x,
        leaderboard[i].position_y);
        window.draw(board_sprite);

        //name
        leaderboard_text.setString(leaderboard[i].username);
        leaderboard_text.setPosition(board_sprite.getPosition() +
        leaderboard_name_offset);
        window.draw(leaderboard_text);

        //score

        leaderboard_text.setString(std::to_string(leaderboard[i].score));
        leaderboard_text.setPosition(board_sprite.getPosition() +
        leaderboard_score_offset);
        window.draw(leaderboard_text);
    }

}

void Toaster::drawToasts(sf::RenderWindow& window, float dt)
{

    for (int i = 0; i < toast_timers.size(); i++)
    {
        toast_timers[i] -= dt;
        if (toast_timers[i] < 0)
        {
            goal_y -= 67;
            toast_timers.erase(toast_timers.begin() + i);
            i--;
        }
    }


    first_toast_position.y = lerp(first_toast_position.y, goal_y,
    0.05);

    int amount = 0;

    v2f toast_position = first_toast_position;
    for(int i = 0; i < toasts.size(); i++)
    {
        toast_slides[i] = lerp(toast_slides[i], 0, 0.3);
        v2f slide_offset(toast_slides[i], 0);
```

```
        toast_sprite.setPosition(toast_position + slide_offset);

        text.setString(toasts[i]);
        text.setPosition(toast_position + text_position +
        slide_offset);

        if (toast_position.y + toast_sprite.getLocalBounds().height -
        24 > 0)
        {
            window.draw(toast_sprite);
            window.draw(text);

            amount++;
        }

        toast_position.y += 67;
    }

}

void Toaster::toast(const string& text)
{
    cout << "Toasting: " << text << "\n";
    toasts.push_back(text);
    toast_slides.push_back(500);
    toast_timers.push_back(lifetime);
}
```