```
#include <SFML/Graphics.hpp>
#include <SFML/System.hpp>
#include <SFML/Audio.hpp>
#include <SFML/Window.hpp>

#include <math.h>
#include <limits.h>
#include <stdint.h>

#include <iostream>
#include <string>
#include <exception>
#include <stdexcept>
#include <vector>

#include "GuitarString.hpp"

#define HZ 44100

vector<int16_t> makeSampleFromString(GuitarString gs) {
vector<int16_t> samples;

    gs.pluck();
    int duration = 8;
    for (int i = 0; i < HZ * duration; i++) {
        gs.tic();
        samples.push_back(gs.sample());
    }
    return samples;
}

int main() {
    Sprite background;
    Texture texture;
    if (!texture.loadFromFile("Keys.png")) {
        cout << "Failed to load background" << endl;
        return EXIT_FAILURE;
    }
    Vector2f backSize(texture.getSize());
    RenderWindow window(VideoMode(1200, 400), "SFML Guitar Hero Lite");
    background.setTexture(texture);

    Vector2f WINSIZE(window.getSize());
    double xScale = (double) WINSIZE.x / backSize.x;
    double yScale = (double) WINSIZE.y / backSize.y;

    background.setScale(xScale, yScale);


    Event event;
    double frequency;
    string keyboard = "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/â\200\231 ";
    vector<vector<int16_t> > samples(37);
    vector<Sound> sounds(37);
    vector<SoundBuffer> soundBuffers(37);

    for(int i = 0; i < 37; i++) {
        frequency = 440 * pow(2, (i - 24) / 12.0);
        GuitarString gs(frequency);
        samples[i] = makeSampleFromString(gs);
        if (!soundBuffers[i].loadFromSamples(&samples[i][0], samples[i].size
```

```
(), 2, HZ))
   62:             throw std::runtime_error("SoundBuffer: failed to load from sampl
es.");
   63:                 sounds[i].setBuffer(soundBuffers[i]);
   64:     }
   65:
   66:     while (window.isOpen()) {
   67:         while (window.pollEvent(event)) {
   68:             switch (event.type) {
   69:             case Event::Closed:
   70:                 window.close();
   71:                 break;
   72:             case Event::TextEntered:
   73:                 if (event.text.unicode < 128) {
   74:                     string temp;
   75:                     temp += static_cast<char>(event.text.unicode);
   76:                     int index = keyboard.find(temp);
   77:                     sounds[index].play();
   78:                 }
   79:                 if (Keyboard::isKeyPressed(Keyboard::Escape)) {
   80:                     window.close();
   81:                     break;
   82:                 }
   83:                 break;
   84:             default:
   85:                 break;
   86:             }
   87:             window.clear();
   88:             window.draw(background);
   89:             window.display();
   90:         }
   91:     }
   92:     return 0;
   93: }
```