**Приложение А**

Листинг программы

|  |  |
| --- | --- |
| //библиотеки для графики и мультимедии  #include <SFML/Graphics.hpp>  #include <SFML/Audio.hpp>  #include <iostream>  #include <vector>  #include <functional>\  //для многопоточности  #include <thread>  //библиотека дял таймеров и анимаций  #include <chrono>  //для поддержки разных языков  #include <locale>  //стандартные алгоритмы  #include <algorithm>  #include <random>  #include <string>  #include <fstream>  #include <cmath>  #include <filesystem>  #ifdef \_WIN32  #include <windows.h>  #endif  //класс для загрузочного окна вставки  class LoadingScreen {  private:  sf::RenderWindow window;  sf::Color backgroundColor;  sf::Color loadingColor;  sf::Font font;  sf::Text title;  sf::Text percentText;  sf::RectangleShape loadingBar;  sf::RectangleShape progressBar;  sf::Sprite logoSprite;  sf::Texture logoTexture;  sf::Sprite backgroundSprite;  sf::Texture backgroundTexture;  bool hasLogo = false;  bool hasBackground = false;  float progress = 0.0f;  bool loadingComplete = false;  sf::Clock clock;  public:  LoadingScreen() : window(sf::VideoMode(800, 600), L"Загрузка приложения", sf::Style::Titlebar | sf::Style::Close) {  #ifdef \_WIN32  SetConsoleOutputCP(CP\_UTF8);  SetConsoleCP(CP\_UTF8);  #endif  std::setlocale(LC\_ALL, "Russian");  std::locale::global(std::locale(""))  // Настройки цветов (светлая тема)  backgroundColor = sf::Color(240, 240, 245);  loadingColor = sf::Color(139, 69, 19); // Коричневый цвет для прогресс-бара  // Настройка окна  window.setFramerateLimit(60);  window.setPosition(sf::Vector2i(  (sf::VideoMode::getDesktopMode().width - 800) / 2,  (sf::VideoMode::getDesktopMode().height - 600) / 2  ));  // Загрузка шрифта  if (!font.loadFromFile("RubikGemstones-Regular.ttf")) {  if (!font.loadFromFile("E:/2 курс/АИП/курсовая/kursovaia/Rubik\_Gemstones/RubikGemstones-Regular.ttf")) { //шрифт  std::wcerr << L"Не удалось загрузить шрифт\n" }  }  // Настройка текста процентов (под прогресс-баром)  percentText.setString(L"0%");  percentText.setFont(font);  percentText.setCharacterSize(40);  percentText.setFillColor(sf::Color(80, 80, 80));  percentText.setStyle(sf::Text::Bold);  // Центрирование текста процентов под прогресс-баром  sf::FloatRect percentBounds = percentText.getLocalBounds();  percentText.setOrigin(percentBounds.width / 2, percentBounds.height / 2);  percentText.setPosition(400, 330); // 30px под прогресс-баром (300+15+15)  }  void centerText(sf::Text& text, float yPos) {  sf::FloatRect bounds = text.getLocalBounds();  text.setOrigin(bounds.width / 2, bounds.height / 2);  text.setPosition(400, yPos); // 400 = 800/2 (центр по горизонтали)  }  // Установка фонового изображения  void setBackgroundImage(const std::string& imagePath) {  if (backgroundTexture.loadFromFile(imagePath)) {  backgroundSprite.setTexture(backgroundTexture);  float scaleX = 800.f / backgroundTexture.getSize().x;  float scaleY = 600.f / backgroundTexture.getSize().y;  backgroundSprite.setScale(scaleX, scaleY);  hasBackground = true;  // Делаем текст более заметным на фоне  title.setFillColor(sf::Color::White);  title.setOutlineThickness(2);  title.setOutlineColor(sf::Color(0, 0, 0, 150));  percentText.setFillColor(sf::Color::White);  percentText.setOutlineThickness(1);  percentText.setOutlineColor(sf::Color(0, 0, 0, 150));  // Перемещаем заголовок выше  centerText(title, 100);  }  else { std::wcerr << L"Не удалось загрузить фоновое изображение\n";  }  }  void setLogo(const std::string& logoPath) {  if (logoTexture.loadFromFile(logoPath)) {  logoSprite.setTexture(logoTexture);  // Центрирование логотипа  sf::FloatRect bounds = logoSprite.getLocalBounds();  logoSprite.setOrigin(bounds.width / 2, bounds.height / 2);  logoSprite.setPosition(400, hasBackground ? 300 : 250);  // Масштабирование если нужно  float maxSize = 200.f;  if (bounds.width > maxSize || bounds.height > maxSize) {  float scale = std::min(maxSize / bounds.width, maxSize / bounds.height);  logoSprite.setScale(scale, scale);  }  hasLogo = true;  centerText(title, hasBackground ? 100 : 150);  }  }  void setTitle(const std::wstring& text) {  title.setString(text);  centerText(title, hasBackground ? 100 : 150);  }  void setColors(sf::Color bgColor, sf::Color loadColor) {  backgroundColor = bgColor;  loadingColor = loadColor;  progressBar.setFillColor(loadColor);  }  void update(float delta) {  if (!loadingComplete) {  progress += delta;  if (progress >= 1.0f) {  progress = 1.0f;  loadingComplete = true;  }  // Плавное заполнение прогресс-бара  float animatedProgress = progress;  if (progress < 0.9f) {  animatedProgress = progress \* 0.9f;  }  progressBar.setSize(sf::Vector2f(400 \* animatedProgress, 15));  percentText.setString(std::to\_wstring(static\_cast<int>(progress \* 100)) + L"%");  centerText(percentText, 450);  }  }  bool isRunning() const {  return window.isOpen();  }  bool isComplete() const {  return loadingComplete;  }  void close() {  // Плавное исчезновение  for (int alpha = 255; alpha >= 0; alpha -= 5) {  title.setFillColor(sf::Color(title.getFillColor().r,  title.getFillColor().g,  title.getFillColor().b,  alpha));  percentText.setFillColor(sf::Color(percentText.getFillColor().r,  percentText.getFillColor().g,  percentText.getFillColor().b,  alpha));  loadingBar.setFillColor(sf::Color(loadingBar.getFillColor().r,  loadingBar.getFillColor().g,  loadingBar.getFillColor().b,  alpha));  progressBar.setFillColor(sf::Color(progressBar.getFillColor().r,  progressBar.getFillColor().g,  progressBar.getFillColor().b,  alpha));  if (hasLogo) {  logoSprite.setColor(sf::Color(255, 255, 255, alpha));  }  if (hasBackground) {  backgroundSprite.setColor(sf::Color(255, 255, 255, alpha));  }  render();  std::this\_thread::sleep\_for(std::chrono::milliseconds(10));  }  window.close();  }  void render() {  window.clear(backgroundColor);  if (hasBackground) {  window.draw(backgroundSprite);  }  if (hasLogo) {  window.draw(logoSprite);  }  window.draw(title);  window.draw(loadingBar);  window.draw(progressBar);  window.draw(percentText);  window.display();  }  void handleEvents() {  sf::Event event;  while (window.pollEvent(event)) {  if (event.type == sf::Event::Closed ||  (event.type == sf::Event::KeyPressed && event.key.code == sf::Keyboard::Escape)) {  window.close();  }  }  }  };  ////////////////////КЛАСС ОСНОВНОГО ПРИЛОЖЕНИЯ/////////////  class MainApp {  private:  sf::RenderWindow window;  sf::Font font;  std::vector<Button> buttons;  sf::Texture backgroundTexture;  sf::Sprite backgroundSprite;  std::vector<sf::Text> buttonLabels;  sf::Music\* gameMusic;  bool isMusicPaused = false;  std::vector<std::string> backgroundPaths = {  "backgrounds/frog.png",  "backgrounds/lev.png",  "backgrounds/osminog.png",  "backgrounds/pchela.png",  "backgrounds/ovca.jpg"  };  std::vector<std::string> backgroundLabels = {  // "Лягушка", "Лев", "Осьминог", "Пчела", "Овца"  };  public:  MainApp(sf::Music\* music) : gameMusic(music), window(sf::VideoMode(800, 600), "Познавайка", sf::Style::Titlebar | sf::Style::Close) {  if (!font.loadFromFile("RubikGemstones-Regular.ttf")) {  if (!font.loadFromFile("Rubik\_Gemstones/RubikGemstones - Regular.ttf")) {  std::cerr << "Не удалось загрузить шрифт\n";  }  }  createButtons();  if (!backgroundTexture.loadFromFile("ovca.jpg")) {  backgroundSprite.setColor(sf::Color(240, 240, 245));  }  else {  backgroundSprite.setTexture(backgroundTexture);  float scaleX = window.getSize().x / (float)backgroundTexture.getSize().x;  float scaleY = window.getSize().y / (float)backgroundTexture.getSize().y;  backgroundSprite.setScale(scaleX, scaleY);  }  }  void createButtons() {  float buttonWidth = 250.f;  float buttonHeight = 60.f;  float startY = 150.f;  float spacing = 20.f;  float rightOffset = 200.f; // Смещение вправо  float centerX = (window.getSize().x - buttonWidth) / 2 + rightOffset;  // Кнопка "Сказки"  buttons.emplace\_back(  sf::Vector2f(buttonWidth, buttonHeight),  sf::Vector2f(centerX, startY),  "Сказки", font, 24,  sf::Color(255, 255, 224), // Голубой фон  sf::Color::Black, // Черный текст  10.f  );  buttons.back().setOnClick([this]() { openFairyTalesWindow(); });  //кнопка для справки  buttons.emplace\_back(  sf::Vector2f(40, 40), // Размер кнопки (40x40)  sf::Vector2f(800 - 40 - 10, 600 - 40 - 10), // Позиция: (750, 550) (правый нижний угол с отступом 10px)  "⚙", font, 20,  sf::Color(200, 200, 200), sf::Color::Black,  10.f  );  buttons.back().setOnClick([this]() {  showSettingsWindow();  });  // Кнопка "Найди совпадения"  buttons.emplace\_back(  sf::Vector2f(buttonWidth, buttonHeight),  sf::Vector2f(centerX, startY + (buttonHeight + spacing)),  "Найди совпадения", font, 24,  sf::Color(255, 255, 224),  sf::Color::Black, // Черный текст  10.f  );  buttons.back().setOnClick([this]() { openFindMatchesWindow(); });  // Кнопка "Словарик"  buttons.emplace\_back(  sf::Vector2f(buttonWidth, buttonHeight),  sf::Vector2f(centerX, startY + 2 \* (buttonHeight + spacing)),  "Словарик", font, 24,  sf::Color(255, 255, 224),  sf::Color::Black, // Черный текст  10.f  );  buttons.back().setOnClick([this]() {  WordGame game;  game.run();  });  }  void openFairyTalesWindow() {  Skazki skazki;  skazki.run();  }  void openFindMatchesWindow() {  FindMatchesGame game;  game.run();  }  void showAboutWindow() {  sf::RenderWindow aboutWindow(sf::VideoMode(500, 400), "Об авторе");  sf::Text aboutText;  // Загрузка текста из файла  std::string aboutContent = loadTextFromFile("E:/2 курс/АИП/курсовая/документация/справка/spravka.chm");  if (aboutContent.empty()) {  aboutContent = "Информация об авторе\n\nРазработчик: ...\nВерсия: 1.0\nГод: 2023";  }  aboutText.setString(aboutContent);  aboutText.setFont(font);  aboutText.setCharacterSize(18);  aboutText.setFillColor(sf::Color::Black);  aboutText.setPosition(20, 20);  while (aboutWindow.isOpen()) {  sf::Event event;  while (aboutWindow.pollEvent(event)) {  if (event.type == sf::Event::Closed) {  aboutWindow.close();  }  }  aboutWindow.clear(sf::Color::White);  aboutWindow.draw(aboutText);  aboutWindow.display();  }  }  std::string loadTextFromFile(const std::string& filename) {  std::ifstream file(filename);  if (file.is\_open()) {  return std::string((std::istreambuf\_iterator<char>(file)),  std::istreambuf\_iterator<char>());  }  return "";  }  void showSettingsWindow() {  sf::RenderWindow settingsWindow(sf::VideoMode(300, 300), "Настройки", sf::Style::Titlebar | sf::Style::Close);  // Создаем кнопки для настроек  float buttonWidth = 200.f;  float buttonHeight = 40.f;  float startY = 50.f;  float spacing = 20.f;  float centerX = (settingsWindow.getSize().x - buttonWidth) / 2;  std::vector<Button> settingsButtons;  // Кнопка "Об авторе"  settingsButtons.emplace\_back(  sf::Vector2f(buttonWidth, buttonHeight),  sf::Vector2f(centerX, startY),  "Об авторе", font, 20,  sf::Color(180, 230, 255), sf::Color::Black,  5.f  );  settingsButtons.back().setOnClick([this, &settingsWindow]() {  settingsWindow.close();  showAboutWindow();  });  // Кнопка "Справка"  settingsButtons.emplace\_back(  sf::Vector2f(buttonWidth, buttonHeight),  sf::Vector2f(centerX, startY + (buttonHeight + spacing)),  "Справка", font, 20,  sf::Color(180, 255, 180), sf::Color::Black,  5.f  );  settingsButtons.back().setOnClick([this, &settingsWindow]() {  settingsWindow.close();  showHelpWindow();  });  // Кнопка "Выбор фона"  settingsButtons.emplace\_back(  sf::Vector2f(buttonWidth, buttonHeight),  sf::Vector2f(centerX, startY + 2 \* (buttonHeight + spacing)),  "Выбор фона", font, 20,  sf::Color(255, 180, 180), sf::Color::Black,  5.f  );  settingsButtons.back().setOnClick([this, &settingsWindow]() {  settingsWindow.close();  openBackgroundSelectionWindow();  });  // Кнопка "Музыка ВКЛ/ВЫКЛ"  settingsButtons.emplace\_back(  sf::Vector2f(buttonWidth, buttonHeight),  sf::Vector2f(centerX, startY + 3 \* (buttonHeight + spacing)),  isMusicPaused ? "Музыка ВКЛ" : "Музыка ВКЛ/ВЫКЛ", font, 20,  sf::Color(255, 255, 180), sf::Color::Black,  5.f  );  settingsButtons.back().setOnClick([this]() {  if (gameMusic) {  if (isMusicPaused) {  gameMusic->play();  }  else {  gameMusic->pause();  }  isMusicPaused = !isMusicPaused;  }  });  while (settingsWindow.isOpen()) {  sf::Event event;  while (settingsWindow.pollEvent(event)) {  if (event.type == sf::Event::Closed) {  settingsWindow.close();  }  for (auto& button : settingsButtons) {  button.handleEvent(event, settingsWindow);  }  }  settingsWindow.clear(sf::Color(240, 240, 245));  // Рисуем заголовок  sf::Text title("Настройки", font, 24);  title.setFillColor(sf::Color::Black);  title.setPosition(centerX, 10);  settingsWindow.draw(title);  // Рисуем кнопки  for (const auto& button : settingsButtons) {  button.draw(settingsWindow);  }  settingsWindow.display();  }  }  void run() {  sf::Clock clock;  while (window.isOpen()) {  sf::Event event;  while (window.pollEvent(event)) {  if (event.type == sf::Event::Closed) {  window.close();  }  for (auto& button : buttons) {  button.handleEvent(event, window);  }  }  for (auto& button : buttons) {  button.update();  }  window.clear();  window.draw(backgroundSprite);  for (const auto& button : buttons) {  button.draw(window);  }  if (event.type == sf::Event::KeyPressed) {  if (event.key.code == sf::Keyboard::P && gameMusic) {  if (isMusicPaused) {  gameMusic->play();  }  else {  gameMusic->pause();  }  isMusicPaused = !isMusicPaused;  }  }  window.display();  }  }  void handleEvents() {  sf::Event event;  while (window.pollEvent(event)) {  if (event.type == sf::Event::Closed) {  window.close();  }  for (auto& button : buttons) {  button.handleEvent(event, window);  }  }  }  void update() {  // Обновление состояния кнопок (например, hover эффект)  for (auto& button : buttons) {  if (button.isMouseOver(window)) {  // Можно добавить эффект при наведении  }  }  }  void render() {  window.clear();  // Отрисовка фона  window.draw(backgroundSprite);  // Отрисовка кнопок  for (const auto& button : buttons) {  button.draw(window);  }  window.display();  }  void openBackgroundSelectionWindow() {  sf::RenderWindow bgWindow(sf::VideoMode(600, 400), "Выбор фона", sf::Style::Titlebar);  std::vector<sf::RectangleShape> colorButtons;  sf::Font font;  if (!font.loadFromFile("kursovaia2/Bubble Sans/BubbleSans-Regular.otf")) {  std::cerr << "Не удалось загрузить шрифт! Попытка использования резервного шрифта..." << std::endl;  if (!font.loadFromFile("kursovaia2/Rubik\_Gemstones/RubikGemstones-Regular.ttf")) {  std::cerr << "Не удалось загрузить резервный шрифт! Текст не будет отображаться." << std::endl;  }  }  // Цвета, соответствующие основным цветам фонов  std::vector<sf::Color> backgroundColors = {  sf::Color(144, 238, 144), // Светло-зеленый (лягушка)  sf::Color(210, 180, 140), // Тан (лев)  sf::Color(32, 178, 170), // синий (осьминог)  sf::Color(135, 206, 235), // (пчела)  sf::Color(255, 255, 255) // Белый (овца)  };  // Создание цветных кнопок  const float buttonWidth = 150.f;  const float buttonHeight = 100.f;  const float margin = 20.f;  const float padding = 30.f;  std::vector<sf::Text> buttonLabels;  for (size\_t i = 0; i < backgroundColors.size(); ++i) {  sf::RectangleShape button(sf::Vector2f(buttonWidth, buttonHeight));  // Позиционирование  float x = margin + (i % 3) \* (buttonWidth + padding);  float y = margin + (i / 3) \* (buttonHeight + padding);  button.setPosition(x, y);  // Настройка внешнего вида  button.setFillColor(backgroundColors[i]);  button.setOutlineThickness(2);  button.setOutlineColor(sf::Color(150, 150, 150));  colorButtons.push\_back(button);  sf::Text label;  label.setFont(font);  label.setString(backgroundLabels[i]);  label.setCharacterSize(16);  label.setFillColor(sf::Color::Black);  float textX = x + (buttonWidth - label.getLocalBounds().width) / 2;  float textY = y + buttonHeight + 5;  label.setPosition(textX, textY);  buttonLabels.push\_back(label);  }  // Кнопка "Закрыть"  Button closeButton(  sf::Vector2f(100, 40),  sf::Vector2f(bgWindow.getSize().x - 110, bgWindow.getSize().y - 50),  "Закрыть", font, 20,  sf::Color(180, 180, 180), sf::Color::Black, 5.f  );  closeButton.setOnClick([&bgWindow]() {  bgWindow.close();  });  // Основной цикл окна  while (bgWindow.isOpen()) {  sf::Event event;  while (bgWindow.pollEvent(event)) {  if (event.type == sf::Event::Closed) {  bgWindow.close();  }  // Обработка кликов по цветным кнопкам  if (event.type == sf::Event::MouseButtonPressed) {  if (event.mouseButton.button == sf::Mouse::Left) {  sf::Vector2f mousePos = bgWindow.mapPixelToCoords(  sf::Vector2i(event.mouseButton.x, event.mouseButton.y));  for (size\_t i = 0; i < colorButtons.size(); ++i) {  if (colorButtons[i].getGlobalBounds().contains(mousePos)) {  // Загружаем соответствующий фон  if (i < backgroundPaths.size()) {  changeBackground(backgroundPaths[i]);  }  bgWindow.close();  break;}}}}  closeButton.handleEvent(event, bgWindow);  }  closeButton.update();  bgWindow.clear(sf::Color(240, 240, 245));  // Заголовок  sf::Text title("Выберите фон", font, 24);  title.setFillColor(sf::Color::Black);  title.setPosition(20, 10);  bgWindow.draw(title);  // Рисуем цветные кнопки  for (const auto& button : colorButtons) {  bgWindow.draw(button);  }  // Кнопка закрытия  closeButton.draw(bgWindow);  bgWindow.display();  }  }  void changeBackground(const std::string& path) {  if (backgroundTexture.loadFromFile(path)) {  backgroundSprite.setTexture(backgroundTexture);  backgroundSprite.setScale(  window.getSize().x / (float)backgroundTexture.getSize().x,  window.getSize().y / (float)backgroundTexture.getSize().y  );  }  else {  backgroundSprite.setColor(sf::Color(240, 240, 245));}}};  /////////////////////////////КЛАСС ДЛЯ ПРОИЗВЕДЕНИЯ СКАЗОК////////  class Skazki {  private:  sf::RenderWindow window;  sf::Texture backgroundTexture;  sf::Sprite backgroundSprite;  sf::Font font;  sf::Music storyMusic;  std::vector<Button> buttons;  std::string labels[4] = {  "Красная Шапочка", "Каша из топора", "Живая шляпа", "Три медведя"  };  std::string audioFiles[4] = {  "skazki/my\_shapochka.mp3",  "skazki/topor.mp3",  "skazki/shlapa.mp3",  "skazki/triMedveda.mp3"  };  public:  Skazki() : window(sf::VideoMode(800, 600), "Сказки", sf::Style::Titlebar | sf::Style::Close) {  // Загрузка фона  if (!backgroundTexture.loadFromFile("skazki/skazkifon.png")) {  std::cerr << "Failed to load background image\n";  backgroundTexture.create(window.getSize().x, window.getSize().y);  backgroundSprite.setColor(sf::Color(240, 240, 245));  }  else {  backgroundSprite.setTexture(backgroundTexture);  float scaleX = window.getSize().x / (float)backgroundTexture.getSize().x;  float scaleY = window.getSize().y / (float)backgroundTexture.getSize().y;  backgroundSprite.setScale(scaleX, scaleY);  }  // Загрузка шрифта  if (!font.loadFromFile("Bubble Sans/BubbleSans-Regular.otf")) {  std::cerr << "Failed to load font!\n";  }  createButtons();  }  void createButtons() {  sf::Vector2f buttonSize(250, 60);  float horizontalSpacing = 80;  float verticalSpacing = 100;  float startX = (window.getSize().x - (buttonSize.x \* 2 + horizontalSpacing)) / 2.0f;  float startY = 200;  for (int i = 0; i < 4; ++i) {  int row = i / 2;  int col = i % 2;  sf::Vector2f position(  startX + col \* (buttonSize.x + horizontalSpacing),  startY + row \* (buttonSize.y + verticalSpacing)  );  Button btn(buttonSize, position, labels[i], font, 20,  sf::Color(176, 224, 230), sf::Color(255, 105, 180));  int index = i; // копия i для безопасного захвата  btn.setOnClick([this, index]() {  playStory(audioFiles[index]);  });  buttons.push\_back(btn);  }  }  void playStory(const std::string& filepath) {  if (storyMusic.getStatus() == sf::SoundSource::Playing) {  storyMusic.stop();  }  if (!storyMusic.openFromFile(filepath)) {  std::cerr << "Error loading " << filepath << "\n";  return;  }  storyMusic.setLoop(false);  storyMusic.setVolume(90);  storyMusic.play();  }  void run() {  while (window.isOpen()) {  sf::Event event;  while (window.pollEvent(event)) {  if (event.type == sf::Event::Closed) {  window.close();  }  for (auto& btn : buttons) {  btn.handleEvent(event, window);  }  }  for (auto& btn : buttons) {  btn.update();  }  window.clear();  window.draw(backgroundSprite);  for (const auto& btn : buttons) {  btn.draw(window);  }  window.display();  }  }  };class Skazki {  private:  sf::RenderWindow window;  sf::Texture backgroundTexture;  sf::Sprite backgroundSprite;  sf::Font font;  sf::Music storyMusic;  std::vector<Button> buttons;  std::string labels[4] = {  "Красная Шапочка", "Каша из топора", "Живая шляпа", "Три медведя"  };  std::string audioFiles[4] = {  "skazki/my\_shapochka.mp3",  "skazki/topor.mp3",  "skazki/shlapa.mp3",  "skazki/triMedveda.mp3"  };  public:  Skazki() : window(sf::VideoMode(800, 600), "Сказки", sf::Style::Titlebar | sf::Style::Close) {  // Загрузка фона  if (!backgroundTexture.loadFromFile("skazki/skazkifon.png")) {  std::cerr << "Failed to load background image\n";  backgroundTexture.create(window.getSize().x, window.getSize().y);  backgroundSprite.setColor(sf::Color(240, 240, 245));  }  else {  backgroundSprite.setTexture(backgroundTexture);  float scaleX = window.getSize().x / (float)backgroundTexture.getSize().x;  float scaleY = window.getSize().y / (float)backgroundTexture.getSize().y;  backgroundSprite.setScale(scaleX, scaleY);  }  // Загрузка шрифта  if (!font.loadFromFile("Bubble Sans/BubbleSans-Regular.otf")) {  std::cerr << "Failed to load font!\n";  }  createButtons();  }  void createButtons() {  sf::Vector2f buttonSize(250, 60);  float horizontalSpacing = 80;  float verticalSpacing = 100;  float startX = (window.getSize().x - (buttonSize.x \* 2 + horizontalSpacing)) / 2.0f;  float startY = 200;  for (int i = 0; i < 4; ++i) {  int row = i / 2;  int col = i % 2;  sf::Vector2f position(  startX + col \* (buttonSize.x + horizontalSpacing),  startY + row \* (buttonSize.y + verticalSpacing)  );  Button btn(buttonSize, position, labels[i], font, 20,  sf::Color(176, 224, 230), sf::Color(255, 105, 180));  int index = i; // копия i для безопасного захвата  btn.setOnClick([this, index]() {  playStory(audioFiles[index]);  });  buttons.push\_back(btn);  }  }  void playStory(const std::string& filepath) {  if (storyMusic.getStatus() == sf::SoundSource::Playing) {  storyMusic.stop();  }  if (!storyMusic.openFromFile(filepath)) {  std::cerr << "Error loading " << filepath << "\n";  return;  }  storyMusic.setLoop(false);  storyMusic.setVolume(90);  storyMusic.play();  }  void run() {  while (window.isOpen()) {  sf::Event event;  while (window.pollEvent(event)) {  if (event.type == sf::Event::Closed) {  window.close();  }  for (auto& btn : buttons) {  btn.handleEvent(event, window);  }  }  for (auto& btn : buttons) {  btn.update();  }  window.clear();  window.draw(backgroundSprite);  for (const auto& btn : buttons) {  btn.draw(window);  }  window.display();  }  }  }; | ////////////////////////////////////////класс игры НАЙДИ СОВПАДЕНИЯ/////////  class FindMatchesGame {  private:  sf::RenderWindow window;  sf::Font font;  sf::Text gameInfoText;  sf::Text winText;  sf::Text titleText;  int currentLevel = 1;  const int MAX\_LEVEL = 2;  sf::Texture backgroundTexture;  sf::Sprite backgroundSprite;  struct Card {  sf::Sprite frontSprite;  sf::Sprite backSprite;  int pairId;  bool isFlipped;  bool isMatched;  float flipProgress;  bool isAnimating;  bool isClosing;  };  std::vector<Card> cards;  std::vector<sf::Texture> textures;  sf::Texture backTexture;  int firstCardIndex = -1;  int secondCardIndex = -1;  int attempts = 0;  int matchedPairs = 0;  bool isWaiting = false;  sf::Clock waitTimer;  const float FLIP\_DURATION = 0.3f;  const float WAIT\_DURATION = 1.0f;  int getPairsCountForLevel(int level) const {  return (level == 1) ? 4 : 6;  }  const float CARD\_WIDTH = 120.f;  const float CARD\_HEIGHT = 150.f;  const float MARGIN = 140.f;  float winAnimationProgress = 0.f;  bool winAnimationActive = false;  const float WIN\_ANIMATION\_DURATION = 2.0f;  public:  FindMatchesGame() : window(sf::VideoMode(800, 700), "Игра 'Найди совпадения'", sf::Style::Titlebar | sf::Style::Close) {  initialize();  }  void initialize() {  window.setSize(sf::Vector2u(800, 700)); // Фиксированный размер  window.setPosition(sf::Vector2i(  (sf::VideoMode::getDesktopMode().width - 800) / 2,  (sf::VideoMode::getDesktopMode().height - 700) / 2  ));  // Настройка окна  window.setVerticalSyncEnabled(true);  // Загрузка шрифтов  if (!font.loadFromFile("RubikGemstones-Regular.ttf")) {  std::cerr << "Failed to load font! Using default." << std::endl;  if (!font.loadFromFile("E:/2 курс/АИП/курсовая/kursovaia/Rubik\_Gemstones/RubikGemstones-Regular.ttf")) {  // Создаем базовый шрифт, если загрузка не удалась  font = sf::Font();  }  }  if (!backgroundTexture.loadFromFile("fongame.png")) {  std::cerr << "Failed to load background texture!\n";  }  else {  backgroundSprite.setTexture(backgroundTexture);  // Растягиваем фон на весь экран  backgroundSprite.setScale(  static\_cast<float>(window.getSize().x) / backgroundTexture.getSize().x,  static\_cast<float>(window.getSize().y) / backgroundTexture.getSize().y  );  }  // Настройка текстовых элементов  setupTextElements();  // Загрузка текстур  loadTextures();  createBackTexture();  // Инициализация игры  resetGame();  }  void setupTextElements() {  gameInfoText.setFont(font);  gameInfoText.setCharacterSize(20);  gameInfoText.setFillColor(sf::Color::Black);  gameInfoText.setPosition(10.f, 10.f);  winText.setFont(font);  winText.setCharacterSize(35);  winText.setFillColor(sf::Color::White);  winText.setStyle(sf::Text::Bold);  titleText.setFont(font);  titleText.setString("Найди совпадения");  titleText.setCharacterSize(30);  titleText.setFillColor(sf::Color::Black);  titleText.setStyle(sf::Text::Bold);  centerText(titleText, 10.f);}  void centerText(sf::Text& text, float yOffset) {  sf::FloatRect bounds = text.getLocalBounds();  text.setPosition(  window.getSize().x / 2 - bounds.width / 2,  yOffset  );  }  void loadTextures() {  int pairs = getPairsCountForLevel(currentLevel);  textures.resize(pairs);  for (int i = 0; i < pairs; ++i) {  std::string path;  if (currentLevel == 1)  path = "figure/card" + std::to\_string(i + 1) + ".jpg";  else  path = "fructies/card" + std::to\_string(i + 1) + ".jpg";  if (!textures[i].loadFromFile(path)) {  sf::Image img;  img.create(100, 150, sf::Color(  rand() % 155 + 100,  rand() % 155 + 100,  rand() % 155 + 100  ));  textures[i].loadFromImage(img);  std::cerr << "Created placeholder for card " << i + 1 << std::endl;  }}}  void createBackTexture() {  if (backTexture.loadFromFile("E:/2 курс/АИП/курсовая/kursovaia/findM/karta.png")) {  return;  }  // Создаем текстуру рубашки, если файл не найден  sf::Image back;  back.create(100, 150, sf::Color(50, 50, 120));  for (int y = 0; y < 150; y += 15) {  for (int x = 0; x < 100; x += 15) {  if ((x / 15 + y / 15) % 2 == 0) {  back.setPixel(x, y, sf::Color(80, 80, 150));  }}}backTexture.loadFromImage(back);  }  void resetGame() {  cards.clear();  firstCardIndex = -1;  secondCardIndex = -1;  attempts = 0;  matchedPairs = 0;  isWaiting = false;  winAnimationActive = false;  int PAIRS\_COUNT = getPairsCountForLevel(currentLevel);  loadTextures();  std::vector<int> pairs;  for (int i = 0; i < PAIRS\_COUNT; ++i) {  pairs.push\_back(i);  pairs.push\_back(i);  }  std::random\_device rd;  std::shuffle(pairs.begin(), pairs.end(), std::mt19937(rd()));  setupCardPositions(pairs);  }  void setupCardPositions(const std::vector<int>& pairs) {  const int COLS = 4;  const int ROWS = (pairs.size() + COLS - 1) / COLS;  const float SPACING\_X = 10.f; //расстояние между колонками  const float SPACING\_Y = 10.f; // убираем расстояние между рядами  for (size\_t i = 0; i < pairs.size(); ++i) {  Card card;  card.pairId = pairs[i];  card.isFlipped = false;  card.isMatched = false;  card.flipProgress = 0.f;  card.isAnimating = false;  card.isClosing = false; if (card.pairId < static\_cast<int>(textures.size())) {  card.frontSprite.setTexture(textures[card.pairId]);  }else {  // Создаем текстуру-заглушку для ошибки  sf::Image errImg;  errImg.create(100, 150, sf::Color::Red);  sf::Texture errTex;  errTex.loadFromImage(errImg);  card.frontSprite.setTexture(errTex);  }// Устанавливаем текстуру для обратной стороны  card.backSprite.setTexture(backTexture);  // Позиционируем карты  int col = i % COLS;  int row = i / COLS;  float posX = MARGIN + col \* (CARD\_WIDTH + SPACING\_X);  float posY = MARGIN + row \* (CARD\_HEIGHT + SPACING\_Y) + 40; //позиция по вертикали  card.frontSprite.setPosition(posX, posY); card.backSprite.setPosition(posX, posY);  // Масштабируем спрайты  scaleSprite(card.frontSprite, CARD\_WIDTH, CARD\_HEIGHT);scaleSprite(card.backSprite, CARD\_WIDTH, CARD\_HEIGHT);  cards.push\_back(card);  }  }  void scaleSprite(sf::Sprite& sprite, float width, float height) {  sf::FloatRect bounds = sprite.getLocalBounds();  sprite.setScale(width / bounds.width, height / bounds.height);  }  void run() {  sf::Clock frameClock;  while (window.isOpen()) {  float deltaTime = frameClock.restart().asSeconds();  processEvents();  update(deltaTime);  render();  }  }  private:  void processEvents() {  sf::Event event;  while (window.pollEvent(event)) {  if (event.type == sf::Event::Closed) {  window.close();  }  if (event.type == sf::Event::KeyPressed && event.key.code == sf::Keyboard::R) {  resetGame();    if (!isWaiting && !winAnimationActive &&  event.type == sf::Event::MouseButtonPressed &&  event.mouseButton.button == sf::Mouse::Left) {  handleClick(event.mouseButton.x, event.mouseButton.y);  }  if (event.type == sf::Event::KeyPressed) {  if (event.key.code == sf::Keyboard::R) {  currentLevel = 1;  resetGame();  }  else if (event.key.code == sf::Keyboard::Enter && winAnimationActive) {  if (currentLevel < MAX\_LEVEL) {  currentLevel++;  winAnimationActive = false;  resetGame();  } }}}}  void handleClick(int x, int y) {  for (size\_t i = 0; i < cards.size(); ++i) {  if (cards[i].isMatched || cards[i].isAnimating) continue;  sf::FloatRect bounds = cards[i].backSprite.getGlobalBounds();  if (bounds.contains(static\_cast<float>(x), static\_cast<float>(y))) {  if (cards[i].isFlipped) continue;  if (firstCardIndex != -1 && secondCardIndex != -1) continue;  startCardFlipAnimation(i);  if (firstCardIndex == -1) {  firstCardIndex = static\_cast<int>(i);  }  else if (secondCardIndex == -1) {  secondCardIndex = static\_cast<int>(i);  attempts++;  } break;}}}  void startCardFlipAnimation(size\_t cardIndex) {  cards[cardIndex].isAnimating = true;  cards[cardIndex].isClosing = false;  cards[cardIndex].flipProgress = 0.f;  }  void update(float deltaTime) {  updateCardAnimations(deltaTime);  if (!isWaiting && firstCardIndex != -1 && secondCardIndex != -1 &&  !cards[firstCardIndex].isAnimating && !cards[secondCardIndex].isAnimating) {  isWaiting = true;  waitTimer.restart();  }  if (isWaiting && waitTimer.getElapsedTime().asSeconds() >= WAIT\_DURATION) {  checkMatch();  isWaiting = false;  }  if (winAnimationActive) {  updateWinAnimation(deltaTime);  }}  void updateCardAnimations(float deltaTime) {  for (auto& card : cards) {  if (card.isAnimating) {  if (card.isClosing) {  card.flipProgress -= deltaTime / FLIP\_DURATION;  if (card.flipProgress <= 0.f) {  card.flipProgress = 0.f;  card.isAnimating = false;  card.isFlipped = false;  }  }  else {  card.flipProgress += deltaTime / FLIP\_DURATION;  if (card.flipProgress >= 1.f) {  card.flipProgress = 1.f;  card.isAnimating = false;  card.isFlipped = true;}}}}}  void updateWinAnimation(float deltaTime) {  winAnimationProgress += deltaTime / WIN\_ANIMATION\_DURATION;  if (winAnimationProgress >= 1.f) {  winAnimationProgress = 1.f;  }  }  void checkMatch() {  if (firstCardIndex == -1 || secondCardIndex == -1) return;  if (cards[firstCardIndex].pairId == cards[secondCardIndex].pairId) {  cards[firstCardIndex].isMatched = true;  cards[secondCardIndex].isMatched = true;  matchedPairs++;  if (matchedPairs == getPairsCountForLevel(currentLevel)) {  startWinAnimation();  }  }  else {  startCardCloseAnimation(firstCardIndex);  startCardCloseAnimation(secondCardIndex);  }  firstCardIndex = -1;  secondCardIndex = -1;  }  void startCardCloseAnimation(int cardIndex) {  cards[cardIndex].isAnimating = true;  cards[cardIndex].isClosing = true;  cards[cardIndex].flipProgress = 1.f;  }  void startWinAnimation() {  winAnimationActive = true;  winAnimationProgress = 0.f;  std::string nextLevelText = (currentLevel < MAX\_LEVEL)  ? "\n\nНажмите Enter для следующего уровня"  : "\n\nНажмите R, чтобы начать заново \n\nНажмите 'Закрыть', чтобы завершить игру";  winText.setString("Уровень " + std::to\_string(currentLevel) + " пройден!\n\nПопытки: " +  std::to\_string(attempts) + nextLevelText);  centerText(winText, window.getSize().y / 2);  }  void render() {  window.clear(sf::Color(230, 230, 250));  window.draw(backgroundSprite);  // Рисуем заголовок  window.draw(titleText);  // Рисуем карты  renderCards();  // Рисуем информацию о игре  renderGameInfo();  // Рисуем анимацию победы, если активна  if (winAnimationActive) {  renderWinAnimation();  }  window.display();  }  void renderCards() {  for (const auto& card : cards) {  if (card.isAnimating) {  renderAnimatedCard(card);  }  else if (card.isFlipped || card.isMatched) {  window.draw(card.frontSprite);  }  else {  window.draw(card.backSprite);  }  }  }  void renderAnimatedCard(const Card& card) {  float scale = 1.0f - std::abs(card.flipProgress - 0.5f) \* 2.0f;  sf::Sprite animSprite;  if (card.flipProgress < 0.5f) {  animSprite = card.backSprite;  }  else {  animSprite = card.frontSprite;  }  animSprite.setScale(  scale \* CARD\_WIDTH / animSprite.getLocalBounds().width,  CARD\_HEIGHT / animSprite.getLocalBounds().height  );  animSprite.setPosition(  card.frontSprite.getPosition().x + (CARD\_WIDTH - CARD\_WIDTH \* scale) / 2,  card.frontSprite.getPosition().y  );  window.draw(animSprite);  }  void renderGameInfo() {  int totalPairs = getPairsCountForLevel(currentLevel);  gameInfoText.setString(  "Уровень: " + std::to\_string(currentLevel) +  "\nПопытки: " + std::to\_string(attempts) +  "\nСовпадения: " + std::to\_string(matchedPairs) + "/" +  std::to\_string(totalPairs) +  "\nНажмите R для перезапуска уровня"  );  window.draw(gameInfoText);  }  void renderWinAnimation() {  sf::RectangleShape overlay(sf::Vector2f(window.getSize().x, window.getSize().y));  overlay.setFillColor(sf::Color(0, 0, 0, static\_cast<sf::Uint8>(150 \* winAnimationProgress)));  window.draw(overlay);  float scale = 0.5f + winAnimationProgress \* 0.3f;  winText.setScale(scale, scale);  winText.setFillColor(sf::Color(255, 255, 255, static\_cast<sf::Uint8>(255 \* winAnimationProgress)));  window.draw(winText);  }  };  ///////////////////КЛАСС ИГРЫ СЛОВАРИК///////////////////////////  class WordGame {  private:  sf::RenderWindow window;  sf::Font font;  sf::RectangleShape skipButtonRect;  sf::Texture backgroundTexture;  sf::Sprite backgroundSprite;  struct GameWord {  std::string word;  std::string imagePath;  };  std::vector<GameWord> dictionary = {  {"машина", "slovarik/car.png"}, {"дом", "slovarik/house.png"},  {"яблоко", "slovarik/apple.png"},{"лев", "slovarik/lev.png"},  {"дерево", "slovarik/tree.png"},{"рыба", "slovarik/ruba.png"},  {"жук", "slovarik/zuk.png"}, {"юла", "slovarik/ula.png"},  {"крот", "slovarik/krot.png"}, {"волк", "slovarik/volk.png"},  {"гусь", "slovarik/gus.png"}, {"кот", "slovarik/kot.png"},  {"горы", "slovarik/goru.png"}, {"мяч", "slovarik/mach.png"}  };  GameWord currentWord;  std::string guessedWord;  std::vector<sf::Text> letterButtons;  sf::Text wordDisplay;  sf::Sprite imageSprite;  sf::Texture imageTexture;  sf::Text skipButton;  sf::Text winCounterText;  int winCounter = 0;  bool showCongratulations = false;  bool showSuperHero = false;  sf::Text congratsText;  sf::Text superHeroText;  sf::Text continueText;  sf::Text menuButtonText;  sf::RectangleShape continueButtonRect;  sf::RectangleShape menuButtonRect;  sf::Clock successTimer;  bool showingSuccess = false;  void loadRandomWord() {  static std::random\_device rd;  static std::mt19937 gen(rd());  std::shuffle(dictionary.begin(), dictionary.end(), gen);  currentWord = dictionary[0];  // Загрузка изображения  if (!imageTexture.loadFromFile("E:/2 курс/АИП/курсовая/kursovaia2/" + currentWord.imagePath)) {  std::cerr << "Error loading image: " << currentWord.imagePath << std::endl;  createPlaceholderTexture();  }  imageSprite.setTexture(imageTexture);  centerImage();  guessedWord.clear();  wordDisplay.setString(guessedWord);  wordDisplay.setFillColor(sf::Color::Black);  createLetterButtons();  updateWinCounter();  }  void createPlaceholderTexture() {  imageTexture.create(200, 200);  sf::Uint8\* pixels = new sf::Uint8[200 \* 200 \* 4];  for (int i = 0; i < 200 \* 200 \* 4; i += 4) {  pixels[i] = 200; // R  pixels[i + 1] = 200; // G  pixels[i + 2] = 200; // B  pixels[i + 3] = 255; // A  }  imageTexture.update(pixels);  delete[] pixels;  }  void centerImage() {  const float targetWidth = 300.f;  const float targetHeight = 280.f;  sf::Vector2u imageSize = imageTexture.getSize();  // Пропорциональное масштабирование  float scale = std::min(targetWidth / imageSize.x, targetHeight / imageSize.y);  imageSprite.setScale(scale, scale);  // Центрируем по центру спрайта  sf::FloatRect localBounds = imageSprite.getLocalBounds();  imageSprite.setOrigin(localBounds.width / 2.f, localBounds.height / 2.f);  // Размещаем строго по центру с вертикальным смещением  imageSprite.setPosition(  window.getSize().x / 2.f, 240.f //чем больше значение, тем ниже картинка на экра  );  }  void createLetterButtons() {  letterButtons.clear();  std::string letters = currentWord.word;  std::string allLetters = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";  std::shuffle(allLetters.begin(), allLetters.end(), std::mt19937(std::random\_device()()));  letters += allLetters.substr(0, 5);  std::shuffle(letters.begin(), letters.end(), std::mt19937(std::random\_device()()));  // Увеличиваем начальную позицию X для сдвига вправо  float startX = 180;  float startY = window.getSize().y – 150; float padding = 10;  float buttonWidth = 35; // Ширина кнопки  float buttonSpacing = 10; // Расстояние между кнопками  for (size\_t i = 0; i < letters.size(); ++i) {  sf::Text letter;  letter.setFont(font);  letter.setString(std::string(1, letters[i]));  letter.setCharacterSize(24);  letter.setFillColor(sf::Color::Black);  sf::FloatRect letterBounds = letter.getLocalBounds();  letter.setOrigin(letterBounds.width / 2, letterBounds.height / 2);  // Изменяем формулу расчета позиции  float xPos = startX + i \* (buttonWidth + buttonSpacing);  // Проверяем, не выходит ли кнопка за границы окна  if (xPos + buttonWidth > window.getSize().x - 100) {  startY += 60;  startX = 100;  xPos = startX;  }  letter.setPosition(xPos + buttonWidth / 2, startY + 15); // Центрируем букву в кнопке  letterButtons.push\_back(letter);  }  skipButton.setFont(font);  skipButton.setString("Пропустить");  skipButton.setCharacterSize(24);  skipButton.setFillColor(sf::Color::Black);  sf::FloatRect bounds = skipButton.getLocalBounds();  skipButton.setOrigin(bounds.width / 2, bounds.height / 2);  skipButton.setPosition(window.getSize().x / 2, window.getSize().y - 40);  skipButtonRect.setSize(sf::Vector2f(bounds.width + 30, bounds.height + 15));  skipButtonRect.setFillColor(sf::Color(255, 218, 185));  skipButtonRect.setOutlineThickness(2);  skipButtonRect.setOutlineColor(sf::Color::Black);  skipButtonRect.setOrigin(skipButtonRect.getSize().x / 2, skipButtonRect.getSize().y / 2);  skipButtonRect.setPosition(skipButton.getPosition());  }  void handleLetterClick(const sf::Vector2f& mousePos) {  if (showSuperHero) {  if (menuButtonRect.getGlobalBounds().contains(mousePos)) {  window.close();  return;  }  }  if (showCongratulations) {  if (continueButtonRect.getGlobalBounds().contains(mousePos)) {  showCongratulations = false;  return;  }  if (menuButtonRect.getGlobalBounds().contains(mousePos)) {  window.close();  return;  }  return;  }  if (showingSuccess) {  return;  }  if (skipButtonRect.getGlobalBounds().contains(mousePos)) {  loadRandomWord();  return;  }  for (const auto& letter : letterButtons) {  sf::FloatRect bounds = letter.getGlobalBounds();  bounds.left -= 20;  bounds.top -= 20;  bounds.width += 40;  bounds.height += 40;  if (bounds.contains(mousePos)) {  guessedWord += letter.getString();  wordDisplay.setString(guessedWord);  checkWord();  break;  }  }  }  void checkWord() {  if (guessedWord.size() == currentWord.word.size()) {  if (guessedWord == currentWord.word) {  wordDisplay.setFillColor(sf::Color::Green);  winCounter++;  updateWinCounter();  if (winCounter == 5) {  showCongratulations = true;  setupCongratulations();  return;  }  else if (winCounter == 10) {  showSuperHero = true;  setupSuperHero();  return;  }  showingSuccess = true;  successTimer.restart();  }  else {  wordDisplay.setFillColor(sf::Color::Red);  guessedWord.clear();  wordDisplay.setString(guessedWord);  }  }  }  void setupCongratulations() {  congratsText.setFont(font);  congratsText.setString("Поздравляем! Вы угадали 5 слов!");  congratsText.setCharacterSize(36);  congratsText.setFillColor(sf::Color::Green);  sf::FloatRect bounds = congratsText.getLocalBounds();  congratsText.setOrigin(bounds.width / 2, bounds.height / 2);  congratsText.setPosition(window.getSize().x / 2, window.getSize().y / 2 - 50);  continueText.setFont(font);  continueText.setString("Продолжить");  continueText.setCharacterSize(24);  continueText.setFillColor(sf::Color::Black);  bounds = continueText.getLocalBounds();  continueText.setOrigin(bounds.width / 2.0f, bounds.height / 2.0f);  continueText.setPosition(window.getSize().x / 2, window.getSize().y / 2 + 50);  continueButtonRect.setSize(sf::Vector2f(bounds.width + 40, bounds.height + 20));  continueButtonRect.setFillColor(sf::Color(200, 200, 200));  continueButtonRect.setOutlineThickness(2);  continueButtonRect.setOutlineColor(sf::Color::Black);  continueButtonRect.setOrigin(continueButtonRect.getSize().x / 2, continueButtonRect.getSize().y / 2);  continueButtonRect.setPosition(continueText.getPosition());  menuButtonText.setFont(font);  menuButtonText.setString("В меню");  menuButtonText.setCharacterSize(24);  menuButtonText.setFillColor(sf::Color::Black);  bounds = menuButtonText.getLocalBounds();  menuButtonText.setOrigin(bounds.width / 2, bounds.height / 2);  menuButtonText.setPosition(window.getSize().x / 2, window.getSize().y / 2 + 100);  menuButtonRect.setSize(sf::Vector2f(bounds.width + 40, bounds.height + 20));  menuButtonRect.setFillColor(sf::Color(200, 200, 200));  menuButtonRect.setOutlineThickness(2);  menuButtonRect.setOutlineColor(sf::Color::Black);  menuButtonRect.setOrigin(menuButtonRect.getSize().x / 2, menuButtonRect.getSize().y / 2);  menuButtonRect.setPosition(menuButtonText.getPosition());  }  void setupSuperHero() {  superHeroText.setFont(font);  superHeroText.setString("Ты новый супер герой!");  superHeroText.setCharacterSize(48);  superHeroText.setFillColor(sf::Color::Yellow);  sf::FloatRect bounds = superHeroText.getLocalBounds();  superHeroText.setOrigin(bounds.width / 2, bounds.height / 2);  superHeroText.setPosition(window.getSize().x / 2, window.getSize().y / 2);  menuButtonText.setFont(font);  menuButtonText.setString("В меню");  menuButtonText.setCharacterSize(24);  menuButtonText.setFillColor(sf::Color::Black);  bounds = menuButtonText.getLocalBounds();  menuButtonText.setOrigin(bounds.width / 2, bounds.height / 2);  menuButtonText.setPosition(window.getSize().x / 2, window.getSize().y / 2 + 100);  menuButtonRect.setSize(sf::Vector2f(bounds.width + 40, bounds.height + 20));  menuButtonRect.setFillColor(sf::Color(200, 200, 200));  menuButtonRect.setOutlineThickness(2);  menuButtonRect.setOutlineColor(sf::Color::Black);  menuButtonRect.setOrigin(menuButtonRect.getSize().x / 2, menuButtonRect.getSize().y / 2);  menuButtonRect.setPosition(menuButtonText.getPosition());  }  void updateWinCounter() {  winCounterText.setFont(font);  winCounterText.setString("Угадано слов: " + std::to\_string(winCounter));  winCounterText.setCharacterSize(20);  winCounterText.setFillColor(sf::Color::Black);  winCounterText.setPosition(20, 20);  }  void drawGameElements() {  window.draw(backgroundSprite);  window.draw(winCounterText);  wordDisplay.setPosition(window.getSize().x / 2, 400);  sf::FloatRect bounds = wordDisplay.getLocalBounds();  wordDisplay.setOrigin(bounds.width / 2, bounds.height / 2);  window.draw(wordDisplay);  for (const auto& letter : letterButtons) {  sf::RectangleShape button(sf::Vector2f(40, 40));  button.setFillColor(sf::Color(240, 240, 240));  button.setOutlineThickness(2);  button.setOutlineColor(sf::Color(180, 180, 180));  button.setPosition(letter.getPosition().x - 20, letter.getPosition().y - 20);  window.draw(button);  window.draw(letter);  }  window.draw(skipButtonRect);  window.draw(skipButton);  window.draw(imageSprite);  }  void drawCongratulations() {  sf::RectangleShape overlay(sf::Vector2f(window.getSize().x, window.getSize().y));  overlay.setFillColor(sf::Color(0, 0, 0, 150));  window.draw(overlay);  window.draw(congratsText);  window.draw(continueButtonRect);  window.draw(continueText);  window.draw(menuButtonRect);  window.draw(menuButtonText);  }  void drawSuperHero() {  sf::RectangleShape overlay(sf::Vector2f(window.getSize().x, window.getSize().y));  overlay.setFillColor(sf::Color(0, 0, 0, 200));  window.draw(overlay);  window.draw(superHeroText);  window.draw(menuButtonRect);  window.draw(menuButtonText);  }  public:  WordGame() : window(sf::VideoMode(800, 600), "Словарик", sf::Style::Titlebar | sf::Style::Close) {  // Загрузка фонового изображения  if (!backgroundTexture.loadFromFile("slovarikfon.png")) {  std::cerr << "Failed to load background image! Using white background." << std::endl;  backgroundTexture.create(window.getSize().x, window.getSize().y);  }  backgroundSprite.setTexture(backgroundTexture);  float scaleX = static\_cast<float>(window.getSize().x) / backgroundTexture.getSize().x;  float scaleY = static\_cast<float>(window.getSize().y) / backgroundTexture.getSize().y;  backgroundSprite.setScale(scaleX, scaleY);  // Загрузка шрифта  if (!font.loadFromFile("palab.ttf")) {  std::cerr << "Failed to load font! Trying fallback font..." << std::endl;  if (!font.loadFromFile("E:/2 курс/АИП/курсовая/kursovaia2/Rubik\_Gemstones/palab.ttf")) {  std::cerr << "Failed to load fallback font! Text will not be displayed." << std::endl;  }  }  wordDisplay.setFont(font);  wordDisplay.setCharacterSize(36);  wordDisplay.setFillColor(sf::Color::Black);  loadRandomWord();  }  void run() {  while (window.isOpen()) {  sf::Event event;  while (window.pollEvent(event)) {  if (event.type == sf::Event::Closed) {  window.close();  }  else if (event.type == sf::Event::MouseButtonPressed) {  if (event.mouseButton.button == sf::Mouse::Left) {  handleLetterClick(window.mapPixelToCoords(  sf::Vector2i(event.mouseButton.x, event.mouseButton.y)));  }  }  }  // Проверяем таймер успеха  if (showingSuccess && successTimer.getElapsedTime().asSeconds() >= 1.5f) {  showingSuccess = false;  loadRandomWord();  }  window.clear(sf::Color::White);  if (showSuperHero) {  drawGameElements();  drawSuperHero();  }  else if (showCongratulations) {  drawGameElements();  drawCongratulations();  }  else {  drawGameElements();  }  window.display();  }  }  }; |