Slot Machine на C++ и SFML

Подробное объяснение кода + вопросы для защиты • дата: 07.02.2026

Что делает программа: открывает окно SFML и отображает 3 "барабана" (текст). Нажатие SPACE делает спин: списывает 200$, выбирает 3 случайных символа из списка и начисляет выигрыш по правилам. Клавиша = показывает общий баланс.

Клавиша	Действие
SPACE (на интро)	Начать игру (убрать экран инструкции)
SPACE (в игре)	Спин: -200$, обновить 3 символа, проверить выигрыш
=	Показать TOTAL (текущий баланс)
Закрытие окна	Завершить программу

1. Структура программы (крупными блоками)

Вся программа находится в main(). Логически она делится на 3 части:

- Инициализация: генератор случайных чисел, окно, шрифты, тексты, линии, переменные баланса.
- Главный цикл: обработка событий (клавиатура/закрытие окна).
- Рендер: очистка кадра, рисование интро или игрового интерфейса, показ кадра.

2. Подключенные библиотеки

В C++ подключение через #include добавляет объявления классов/функций из библиотек.

Файл	Зачем нужен
SFML/Graphics.hpp	Окно, текст, цвета, фигуры, draw(), события
vector	Контейнер std::vector (список элементов с размером)
cstdlib	rand(), srand()
ctime	time() для зерна генератора
string	std::string и работа со строками

3. Инициализация случайных чисел

Строка:

```
std::srand(static_cast<unsigned>(std::time(nullptr)));
```

Зачем: функция rand() выдает псевдослучайные числа. Если не вызвать srand(...), то при каждом запуске игры последовательность будет одинаковой.

Почему time(nullptr): текущее время меняется каждую секунду - это удобное "зерно" (seed).

Почему static_cast<unsigned>: srand ожидает unsigned int. time() возвращает time_t (тип зависит от системы), поэтому делаем явное приведение.

4. Создание окна

```
sf::RenderWindow window(sf::VideoMode({ 900u, 500u }), "Slot Machine");
```

RenderWindow - главный объект SFML: хранит окно, принимает события и позволяет рисовать.

VideoMode задает размер окна. Суффикс u означает unsigned (без знака).

5. Шрифт и текст

```
sf::Font font;
font.openFromFile("C:/Windows/Fonts/arial.ttf");
```

В SFML текст (sf::Text) отображается только если к нему привязан шрифт (sf::Font).

Улучшение для защиты: добавить проверку загрузки шрифта и обработать ошибку (иначе на другом ПК путь может не совпасть).

```
if (!font.openFromFile("C:/Windows/Fonts/arial.ttf"))
    return 1;
```

6. Переменные игры (баланс и стоимость)

```
int balance = 0;
const int spinCost = 200;
```

balance - текущие деньги игрока. spinCost - цена спина. const защищает от случайного изменения.

7. Данные "барабанов": список символов и три надписи

Список возможных символов:

```
std::vector<std::string> fruitNames = { "Cherry", "Lemon", "Bell", "777" };
```

Три барабана - это три объекта sf::Text в векторе slotText.

Цикл создает 3 текста, задаёт им шрифт, стартовую строку, размер и цвет:

```
std::vector<sf::Text> slotText;
for (int i = 0; i < 3; i++)
{
    sf::Text t(font);
    t.setString(fruitNames[i]);
    t.setCharacterSize(40);
    t.setFillColor(sf::Color::White);
    slotText.push_back(t);
}
```

Почему vector: удобно хранить несколько одинаковых объектов и рисовать их циклом.

Дальше ты ставишь их на экран по координатам:

```
slotText[0].setPosition({150.f, 200.f});
slotText[1].setPosition({400.f, 200.f});
slotText[2].setPosition({650.f, 200.f});
```

Координаты - float (числа с точкой), потому что SFML использует float для позиционирования.

8. Визуальные разделители: линии

```
sf::RectangleShape line1({2.f, 500.f});
sf::RectangleShape line2({2.f, 500.f});
line1.setPosition({300.f, 0.f});
line2.setPosition({600.f, 0.f});
line1.setFillColor(sf::Color::White);
line2.setFillColor(sf::Color::White);
```

RectangleShape - прямоугольник. Если ширина 2 пикселя, получается тонкая вертикальная линия. Ты ставишь две линии, чтобы визуально разделить 3 колонки.

9. Тексты интерфейса: баланс и результат

scoreText - показывает баланс, resultText - показывает результат спина (WIN/LOSE/JACKPOT/TOTAL).

```
sf::Text scoreText(font);
scoreText.setString("Balance: 0$");
scoreText.setCharacterSize(22);
scoreText.setFillColor(sf::Color{150, 255, 0});
scoreText.setPosition({700.f, 20.f});

sf::Text resultText(font);
resultText.setString("");
resultText.setCharacterSize(26);
resultText.setPosition({350.f, 50.f});
```

Цвет результата меняется: оранжевый при выигрыше, красный при проигрыше.

10. Intro-экран (инструкция)

Переменная intro включает режим приветствия:

```
bool intro = true;
sf::Text introText(font);
introText.setString("Welcome ...");
...
```

Пока intro=true, ты рисуешь только текст инструкции и не даешь крутить слот. Нажимаем SPACE - меняем intro на false.

11. Главный цикл приложения

В графических программах обычно есть цикл, который работает до закрытия окна. У тебя это:

```
while (window.isOpen())
{
    while (auto event = window.pollEvent())
    {
        ...
    }

    window.clear(...);
    ...
    window.display();
}
```

Каждый проход цикла - один "кадр":

1. Сначала обрабатываем все события (клавиши, закрытие окна).
2. Потом очищаем экран (clear).
3. Рисуем нужные объекты (draw).
4. Показываем кадр (display).

12. Обработка событий (event loop)

SFML хранит события в очереди. pollEvent() вытаскивает их по одному:

```
while (auto event = window.pollEvent())
{
    if (event->is<sf::Event::Closed>())
        window.close();

    if (event->is<sf::Event::KeyPressed>())
    {
        auto key = event->getIf<sf::Event::KeyPressed>();
        ...
    }
}
```

Closed - пользователь нажал крестик. Мы закрываем окно.

KeyPressed - нажата клавиша. key->code хранит код клавиши.

13. Переключение режимов: интро -> игра

```
if (intro && key->code == sf::Keyboard::Key::Space)
    intro = false;
```

Это очень типичный приём: одна переменная состояния (intro) переключает, что рисовать и какие действия разрешены.

14. Спин (SPACE в игре): деньги, случайность, обновление UI

Когда intro уже выключен и нажата SPACE, происходит спин:

```
if (key->code == sf::Keyboard::Key::Space)
{
    balance -= spinCost;

    int r1 = rand() % 4;
    int r2 = rand() % 4;
    int r3 = rand() % 4;

    slotText[0].setString(fruitNames[r1]);
    slotText[1].setString(fruitNames[r2]);
    slotText[2].setString(fruitNames[r3]);
    ...
}
```

1) Сначала списываешь стоимость: balance -= 200.

2) Генерируешь три индекса 0..3 (потому что 4 символа).

3) По этим индексам обновляешь текст барабанов.

Важно: сейчас можно уйти в минус. На защите можно сказать, что это легко исправляется проверкой balance >= spinCost.

```
if (balance < spinCost)
{
    resultText.setString("Not enough money!");
    resultText.setFillColor(sf::Color::Red);
    return; // или просто не делать спин
}
```

15. Проверка выигрыша

Ты проверяешь комбинации по индексам r1, r2, r3.

15.1 Джекпот (777)

```
if (r1 == 3 && r2 == 3 && r3 == 3)
{
    balance += 10000;
    resultText.setString("JACKPOT 777!");
    resultText.setFillColor(sf::Color(255, 165, 0));
}
```

Почему 3? Потому что в fruitNames "777" - это 4-й элемент (индекс 3).

15.2 Обычный выигрыш (любые 3 одинаковые)

```
else if (r1 == r2 && r2 == r3)
{
    balance += 1000;
    resultText.setString("YOU WIN!");
    resultText.setFillColor(sf::Color(255, 165, 0));
}
```

Проверка r1==r2 и r2==r3 гарантирует, что все три равны.

15.3 Проигрыш

```
else
{
    resultText.setString("YOU LOSE");
    resultText.setFillColor(sf::Color::Red);
}
```

Почему джекпот вынесен отдельно: он тоже "3 одинаковых", но приз другой. Если бы ты не выделил его первым, он попал бы в обычный выигрыш +1000.

16. Обновление табло баланса

```
scoreText.setString("Balance: " + std::to_string(balance) + "$");
```

std::to_string превращает число в строку. Потом строка собирается как "Balance: 123$".

17. Клавиша '=': показать TOTAL

```
if (key->code == sf::Keyboard::Key::Equal)
{
    resultText.setString("TOTAL: " + std::to_string(balance) + "$");
    resultText.setFillColor(sf::Color::White);
}
```

Это просто дополнительная команда: выводит текущий баланс другим текстом.

18. Рисование (render loop)

```
window.clear(sf::Color::Black);

if (intro)
    window.draw(introText);
else
{
    window.draw(line1);
    window.draw(line2);

    for (auto& t : slotText)
        window.draw(t);

    window.draw(scoreText);
    window.draw(resultText);
}

window.display();
```

clear очищает экран. Потом ты рисуешь либо интро, либо игру. display показывает готовый кадр.

Цикл for (auto& t : slotText) рисует все 3 барабана без повторения кода.

19. Полный листинг кода (как в проекте)

Ниже - исходный код целиком (для удобства защиты).

```
#include <SFML/Graphics.hpp>
#include <vector>
#include <cstdlib>
#include <ctime>
#include <string>

int main()
{
    std::srand(static_cast<unsigned>(std::time(nullptr)));
    // ================= WINDOW =================
```

```
sf::RenderWindow window(sf::VideoMode({ 900u, 500u }), "Slot Machine");

// ================= FONT =================
sf::Font font;
font.openFromFile("C:/Windows/Fonts/arial.ttf");

// ================= MONEY =================
int balance = 0;
const int spinCost = 200;

// ================= FRUITS TEXT =================
std::vector<std::string> fruitNames = { "Cherry", "Lemon", "Bell", "777" };
std::vector<sf::Text> slotText;
for (int i = 0; i < 3; i++)
{
    sf::Text t(font);             // SFML 3: конструктор с font
    t.setString(fruitNames[i]);
    t.setCharacterSize(40);
    t.setFillColor(sf::Color::White);
    slotText.push_back(t);
}

slotText[0].setPosition(sf::Vector2f{ 150.f, 200.f });
slotText[1].setPosition(sf::Vector2f{ 400.f, 200.f });
slotText[2].setPosition(sf::Vector2f{ 650.f, 200.f });

// ================= LINES =================
sf::RectangleShape line1(sf::Vector2f{ 2.f, 500.f });
sf::RectangleShape line2(sf::Vector2f{ 2.f, 500.f });
line1.setPosition(sf::Vector2f{ 300.f, 0.f });
line2.setPosition(sf::Vector2f{ 600.f, 0.f });
line1.setFillColor(sf::Color::White);
line2.setFillColor(sf::Color::White);

// ================= SCOREBOARD =================
sf::Text scoreText(font);
scoreText.setString("Balance: 0$");
scoreText.setCharacterSize(22);
scoreText.setFillColor(sf::Color{ 150, 255, 0 });
scoreText.setPosition(sf::Vector2f{ 700.f, 20.f });

sf::Text resultText(font);
resultText.setString("");
resultText.setCharacterSize(26);
resultText.setPosition(sf::Vector2f{ 350.f, 50.f });

// ================= INTRO SCREEN =================
bool intro = true;
sf::Text introText(font);
introText.setString(
    "Welcome to the Slot Machine!\n"
    "Press SPACE to start.\n"
    "Spin costs 200$.\n"
    "777 = 10000$\n"
    "3 same fruits = 1000$\n"
    "= shows total balance"
);
introText.setCharacterSize(22);
introText.setFillColor(sf::Color::White);
introText.setPosition(sf::Vector2f{ 100.f, 150.f });

// ================= MAIN LOOP =================
while (window.isOpen())
{
    while (auto event = window.pollEvent())
```

```
{
    if (event->is<sf::Event::Closed>())
        window.close();

    if (event->is<sf::Event::KeyPressed>())
    {
        auto key = event->getIf<sf::Event::KeyPressed>();

        if (intro && key->code == sf::Keyboard::Key::Space)
            intro = false;

        else if (!intro)
        {
            if (key->code == sf::Keyboard::Key::Space)
            {
                balance -= spinCost;

                int r1 = rand() % 4;
                int r2 = rand() % 4;
                int r3 = rand() % 4;

                slotText[0].setString(fruitNames[r1]);
                slotText[1].setString(fruitNames[r2]);
                slotText[2].setString(fruitNames[r3]);

                if (r1 == 3 && r2 == 3 && r3 == 3)
                {
                    balance += 10000;
                    resultText.setString("JACKPOT 777!");
                    resultText.setFillColor(sf::Color(255, 165, 0));
                }
                else if (r1 == r2 && r2 == r3)
                {
                    balance += 1000;
                    resultText.setString("YOU WIN!");
                    resultText.setFillColor(sf::Color(255, 165, 0));
                }
                else
                {
                    resultText.setString("YOU LOSE");
                    resultText.setFillColor(sf::Color::Red);
                }

                scoreText.setString("Balance: " + std::to_string(balance) + "$");
            }

            if (key->code == sf::Keyboard::Key::Equal)
            {
                resultText.setString("TOTAL: " + std::to_string(balance) + "$");
                resultText.setFillColor(sf::Color::White);
            }
        }
    }
}

window.clear(sf::Color::Black);

if (intro)
    window.draw(introText);
else
{
    window.draw(line1);
    window.draw(line2);

    for (auto& t : slotText)
```

```
                window.draw(t);

            window.draw(scoreText);
            window.draw(resultText);
        }

        window.display();
    }

    return 0;
}
```

20. Частые вопросы на защите и готовые ответы

1. Зачем нужно srand(time(nullptr))?

Чтобы rand() давал разные результаты при каждом запуске. Без srand последовательность будет одинаковой каждый раз.

> Как ответить уверенно: Чтобы rand() давал разные результаты при каждом запуске. Без srand последовательность будет одинаковой каждый раз.

2. Почему rand() % 4, а не другое число?

Потому что в fruitNames 4 элемента. Индексы идут 0..3. Остаток от деления на 4 дает ровно этот диапазон.

> Как ответить уверенно: Потому что в fruitNames 4 элемента. Индексы идут 0..3. Остаток от деления на 4 дает ровно этот диапазон.

3. Почему джекпот проверяется отдельным if перед проверкой "3 одинаковых"?

Потому что 777-777-777 тоже три одинаковых. Если не проверить первым, джекпот попадет в обычный выигрыш и даст +1000 вместо +10000.

> Как ответить уверенно: Потому что 777-777-777 тоже три одинаковых. Если не проверить первым, джекпот попадет в обычный выигрыш и даст +1000 вместо +10000.

4. Что делает главный цикл while(window.isOpen())?

Это цикл жизни приложения. Пока окно открыто, программа повторяет: обработка событий -> рисование кадра.

> Как ответить уверенно: Это цикл жизни приложения. Пока окно открыто, программа повторяет: обработка событий -> рисование кадра.

5. Зачем pollEvent() внутри второго while?

События (клавиши/закрытие) лежат в очереди. pollEvent() вытаскивает их по одному, пока очередь не пустая.

> Как ответить уверенно: События (клавиши/закрытие) лежат в очереди. pollEvent() вытаскивает их по одному, пока очередь не пустая.

6. Почему сначала window.clear(), а потом draw() и display()?

clear очищает кадр. draw рисует новый кадр. display показывает готовый кадр. Иначе были бы артефакты (следы прошлого кадра).

> Как ответить уверенно: clear очищает кадр. draw рисует новый кадр. display показывает готовый кадр. Иначе были бы артефакты (следы прошлого кадра).

7. Почему slotText - это vector, а не три отдельные переменные?

Так удобнее: можно хранить все барабаны вместе и рисовать/обновлять в цикле. Код короче и проще поддерживать.

> Как ответить уверенно: Так удобнее: можно хранить все барабаны вместе и рисовать/обновлять в цикле. Код короче и проще поддерживать.

8. Что такое sf::Text и sf::Font и почему они оба нужны?

sf::Font хранит шрифт (данные букв), а sf::Text - объект, который рисуется. Text использует Font, чтобы знать как выглядят символы.

> Как ответить уверенно: sf::Font хранит шрифт (данные букв), а sf::Text - объект, который рисуется. Text использует Font, чтобы знать как выглядят символы.

9. Что будет, если шрифт не загрузится (путь неверный)?

Сейчас в коде нет проверки, поэтому текст может не отобразиться. Улучшение: if(!font.openFromFile(...)) обработать ошибку.

> Как ответить уверенно: Сейчас в коде нет проверки, поэтому текст может не отобразиться. Улучшение: if(!font.openFromFile(...)) обработать ошибку.

10. Почему координаты позиции float (150.f), а не int?

SFML использует float-координаты. Это дает точное позиционирование и плавность при анимации.

> Как ответить уверенно: SFML использует float-координаты. Это дает точное позиционирование и плавность при анимации.

11. Можно ли уйти в минус по балансу?

Да, сейчас можно. Это легко исправить проверкой balance >= spinCost и выводом сообщения "Not enough money".

> Как ответить уверенно: Да, сейчас можно. Это легко исправить проверкой balance >= spinCost и выводом сообщения "Not enough money".

12. Почему используешь std::to_string?

Чтобы превратить число balance в строку и собрать текст для scoreText/resultText.

> Как ответить уверенно: Чтобы превратить число balance в строку и собрать текст для scoreText/resultText.

13. Как работает логика выигрыша "r1==r2 && r2==r3"?

Если r1 равен r2 и r2 равен r3, значит все три одинаковые (транзитивность равенства).

> Как ответить уверенно: Если r1 равен r2 и r2 равен r3, значит все три одинаковые (транзитивность равенства).

14. Что делает клавиша '=' в игре?

Она не меняет баланс, а только выводит сообщение TOTAL с текущим балансом.

> Как ответить уверенно: Она не меняет баланс, а только выводит сообщение TOTAL с текущим балансом.

15. Какие улучшения ты бы сделал, если было больше времени?

Проверка баланса перед спином; проверка загрузки шрифта; замена rand на ; анимация прокрутки; вынесение логики в функции; настройка вероятностей (777 реже).

> Как ответить уверенно: Проверка баланса перед спином; проверка загрузки шрифта; замена rand на ; анимация прокрутки; вынесение логики в функции; настройка вероятностей (777 реже).

21. Мини-список "если придираются" (сильные замечания)

- Проверка шрифта: путь к Arial может отличаться на другом ПК. Лучше хранить шрифт рядом с .exe и грузить относительным путём.
- Ограничение баланса: не давать спин, если денег меньше 200.
- rand()%4 - простой вариант. Более правильный - (mt19937 + uniform_int_distribution).
- Разделить проект на функции: spin(), checkWin(), updateTexts(), drawScene(). Это улучшит читаемость и защиту.
- Добавить стартовый баланс (например 1000$) или кнопку пополнения - чтобы игрок мог играть сразу.