Общие требования Игра должна иметь полноценную звуковую систему с фоновой музыкой и звуковыми эффектами.

Все звуки должны корректно загружаться и воспроизводиться в современных браузерах.

У игрока должна быть возможность включать/выключать звуки и музыку через меню.

Запуск игры должен происходить только после нажатия Enter, с проигрыванием звука старта.

Функциональные требования

2.1. Звуковая система (SoundManager) Инициализация звуков:

Все звуки должны загружаться при старте игры.

Если звук не загрузился, игра должна продолжать работу без него (без критических ошибок).

Поддержка Web Audio API с fallback на HTML5 Audio.

init() – загрузка звуков.

playSound(name) – воспроизведение звука (с обработкой ошибок).

toggleSound() – вкл/выкл звуковые эффекты.

toggleMusic() – вкл/выкл фоновую музыку.

startMusic() – запуск музыки (с автоповтором).

2.2. Меню звуков Элементы интерфейса:

Кнопка "Звук: ВКЛ/ВЫКЛ" – управление звуковыми эффектами.

Кнопка "Музыка: ВКЛ/ВЫКЛ" – управление фоновой музыкой.

Кнопка "Играть" – Чтобы выбрать уровень.

Визуальное оформление:

2.3. GameManager (логика игры) Запуск игры:

Игра начинается только после нажатия Enter.

При нажатии проигрывается звук start.mp3.

Если звук не загружен – игра стартует без него.

Управление паузой:

Escape – пауза/продолжение игры.

При паузе музыка приостанавливается.

Обработка завершения игры:

При победе – victory.mp3.

При поражении – gameOver.mp3.

Тестирование

3.1. Проверка звуков Все звуки должны загружаться без ошибок 404.

Громкость должна соответствовать требованиям.

Музыка должна зацикливаться.

3.2. Проверка меню Кнопки звука/музыки должны корректно переключать состояния.

Нажатие Enter должно запускать игру с звуком.

3.3. Проверка GameManager При победе/поражении должны играть соответствующие звуки.

**Движок игры :**

**Как устроен движок?**

1. Получает canvas и его контекст для рисования

2. Устанавливает размеры игрового поля (35x50 блоков)

3. Создает три типа блоков:

4. Красные (разрушаемые, 10 очков)

5. Зеленые (центральные, 20 очков)

6. Серые (крест, неразрушаемые)

**1. Очистка кадра**

\* ctx.clearRect(0, 0, canvas.width, canvas.height) - Стирает предыдущий кадр

**2. Отрисовка объектов:**

\* Блоки (drawBlocks())

\* Мяч (drawBall())

\* Платформа (drawPaddle())

**3. Физика и управление:**

**Пояснение:**

\* movePlatform() - двигает платформу по нажатию клавиш

\* moveBall() - обрабатывает:

\* Движение мяча (ball.x += dx, ball.y += dy)

\* Отскоки от стен(об этом ниже)

\* Столкновения с блоками и платформой

**4. Система столкновений**

function checkBlockCollision(block) {

      const blockLeft = block.x \* blockSize;

      const blockRight = blockLeft + blockSize;

      const blockTop = block.y \* blockSize;

      const blockBottom = blockTop + blockSize;

      return (

        ball.x + ball.radius > blockLeft &&

        ball.x - ball.radius < blockRight &&

        ball.y + ball.radius > blockTop &&

        ball.y - ball.radius < blockBottom

      );

    }

Проверяет пересечение мяча с каждым блоком

**5. Игровая логика:**

При разрушении блока:

\* Увеличивает счет (score += 10/20)

for (let i = redBlocks.length - 1; i >= 0; i--) {

        const block = redBlocks[i];

        if (block.status && checkBlockCollision(block)) {

          ball.dy = -ball.dy;

          block.status = false;

          score += 10;

          scoreSpan.textContent = score;

          break;

        }

      }

\* Меняет направление мяча

 // Столкновение с платформой

      if(

        ball.y + ball.radius > platform.y &&

        ball.x > platform.x &&

        ball.x < platform.x + platform.width

      ) {

        ball.dy = -ball.speed;

        // Изменение направления по месту удара

        const hitPosition = (ball.x - (platform.x + platform.width / 2)) / (platform.width / 2);

        ball.dx = hitPosition \* ball.speed;

      }

При потере мяча:

\* Уменьшает жизни (lives--)

if(ball.y + ball.radius > canvas.height) {

        lives--;

        livesSpan.textContent = lives;

        if(lives <= 0) {

          gameOver();

        } else {

          resetBall();

        }

      }

\* Сбрасывает мяч (resetBall())

function resetBall() {

  ball.x = canvas.width / 2;

  ball.y = canvas.height - 60;

  ball.dx = 2 \* (Math.random() > 0.5 ? 1 : -1); // тоже изменено с 3 на 2

  ball.dy = -2; // изменено с -3 на -2

  platform.x = canvas.width / 2 - platform.width / 2;

}

**6. Особенности управления:**

\* Клавиши A/D двигают платформу

// Обработка клавиш (A/D)

    document.addEventListener('keydown', (e) => {

      if(e.key === 'd' || e.key === 'D') {  // Вправо - D

        platform.dx = platform.speed;

      } else if(e.key === 'a' || e.key === 'A') {  // Влево - A

        platform.dx = -platform.speed;

      }

    });

    document.addEventListener('keyup', (e) => {

      if(e.key === 'd' || e.key === 'D' || e.key === 'a' || e.key === 'A') {

        platform.dx = 0;

      }

    });

\* Угол отскока мяча зависит от места удара о платформу:

const hitPosition = (ball.x - (platform.x + platform.width / 2)) / (platform.width / 2);

        ball.dx = hitPosition \* ball.speed;

**7. Оптимизация:**

\* Использует requestAnimationFrame для плавной анимации (~60 FPS)

\* Перерисовывает только изменившиеся элементы

**8. Условия завершения:**

Победа: checkWin() возвращает true, когда не осталось красных/зеленых блоков

function checkWin() {

      const redLeft = redBlocks.some(block => block.status);

      const greenLeft = greenCenter.some(block => block.status);

      return !redLeft && !greenLeft;

    }

Поражение: при lives <= 0

if(lives <= 0) {

          gameOver();

        } else {

          resetBall();

        }