GUI с JavaFX

Лекция 8: Создание пользовательского интерфейса

Преподаватель: [Ваше имя]

Группа: 203

Семестр: Осенний 2024

План лекции

1. **Введение в JavaFX**
2. **Основные компоненты**
3. **Создание интерфейса**
4. **Обработка событий**
5. **Анимации и эффекты**
6. **Практический пример: Игровой интерфейс**

Введение в JavaFX

Что такое JavaFX?

JavaFX — современная платформа для создания настольных приложений с богатым пользовательским интерфейсом.

Преимущества:

- **Современный дизайн** и стили
- **FXML** для разделения логики и интерфейса
- **CSS** для стилизации
- **Встроенные анимации** и эффекты
- **Кроссплатформенность**

Архитектура JavaFX

Основные компоненты:

```
Application (главный класс)
├── Stage (окно приложения)
├── Scene (сцена)
├── Node (элементы интерфейса)
│   ├── Control (управляющие элементы)
│   ├── Shape (геометрические фигуры)
│   └── Container (контейнеры)
└── Event (события)
```

Жизненный цикл:

1. **start()** — инициализация приложения
2. **Создание Stage и Scene**

Основной класс приложения

```
public class GameApplication extends Application {

    @Override
    public void start(Stage primaryStage) {
        // Создание главного окна
        primaryStage.setTitle("Пошаговая стратегия");
        primaryStage.setWidth(1200);
        primaryStage.setHeight(800);

        // Создание корневого элемента
        BorderPane root = new BorderPane();

        // Создание сцены
        Scene scene = new Scene(root);

        // Установка сцены в окно
        primaryStage.setScene(scene);

        // Отображение окна
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Основные компоненты интерфейса

Контейнеры:

- **BorderPane** — разметка по границам
- **HBox/VBox** — горизонтальная/вертикальная компоновка
- **GridPane** — табличная разметка
- **StackPane** — наложение элементов
- **TabPane** — вкладки

Управляющие элементы:

- **Button** — кнопки
- **Label** — метки
- **TextField** — текстовые поля

BorderPane разметка

```
BorderPane root = new BorderPane();

// Верхняя панель (меню)
HBox topBar = new HBox(10);
topBar.setPadding(new Insets(10));
topBar.getChildren().addAll(
    new Button("Новая игра"),
    new Button("Загрузить"),
    new Button("Сохранить"),
    new Button("Настройки")
);
root.setTop(topBar);

// Левая панель (информация)
VBox leftPanel = new VBox(5);
leftPanel.setPadding(new Insets(10));
leftPanel.getChildren().addAll(
    new Label("Игрок: Player1"),
    new Label("Уровень: 5"),
    new Label("Золото: 1000")
);
root.setLeft(leftPanel);

// Центральная область (игровое поле)
root.setCenter(createGameBoard());

// Правая панель (действия)
root.setRight(createActionPanel());
```

Создание игрового поля

```
private GridPane createGameBoard() {
    GridPane gameBoard = new GridPane();
    gameBoard.setGridLinesVisible(true);
    gameBoard.setHgap(2);
    gameBoard.setVgap(2);

    // Создание клеток 10x10
    for (int row = 0; row < 10; row++) {
        for (int col = 0; col < 10; col++) {
            Rectangle cell = new Rectangle(60, 60);
            cell.setFill(Color.LIGHTGREEN);
            cell.setStroke(Color.DARKGREEN);

            // Обработчик кликов
            final int finalRow = row;
            final int finalCol = col;
            cell.setOnMouseClicked(e -> handleCellClick(finalRow, finalCol));

            gameBoard.add(cell, col, row);
        }
    }

    return gameBoard;
}

private void handleCellClick(int row, int col) {
    System.out.println("Клик по клетке: " + row + ", " + col);
    // Логика обработки клика
}
```

Обработка событий

Типы событий:

- **MouseEvent** — события мыши
- **KeyEvent** — события клавиатуры
- **ActionEvent** — события действий
- **WindowEvent** — события окна

Обработчики событий:

```
Button attackButton = new Button("Атаковать");
attackButton.setOnAction(e -> handleAttack());

ComboBox<String> unitType = new ComboBox<>();
unitType.getItems().addAll("Warrior", "Archer", "Mage");
unitType.setOnAction(e -> handleUnitSelection());
```

Создание панели действий

```
private VBox createActionPanel() {
    VBox actionPanel = new VBox(10);
    actionPanel.setPadding(new Insets(10));
    actionPanel.setPrefWidth(200);

    // Информация о выбранном юните
    Label unitInfo = new Label("Выберите юнита");
    unitInfo.setWrapText(true);
    actionPanel.getChildren().add(unitInfo);

    // Кнопки действий
    Button moveButton = new Button("Двигаться");
    moveButton.setOnAction(e -> handleMoveAction());
    moveButton.setMaxWidth(Double.MAX_VALUE);

    Button attackButton = new Button("Атаковать");
    attackButton.setOnAction(e -> handleAttackAction());
    attackButton.setMaxWidth(Double.MAX_VALUE);

    Button buildButton = new Button("Строить");
    buildButton.setOnAction(e -> handleBuildAction());
    buildButton.setMaxWidth(Double.MAX_VALUE);

    actionPanel.getChildren().addAll(moveButton, attackButton, buildButton);

    return actionPanel;
}
```

FXML для разделения логики и интерфейса

game_interface.fxml:

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.control.*?>
<?import javafx.geometry.Insets?>

<BorderPane xmlns="http://javafx.com/javafx"
            xmlns:fx="http://javafx.com/fxml"
            fx:controller="game.GameController">

    <top>
        <HBox spacing="10" padding="10">
            <Button text="Новая игра" onAction="#handleNewGame"/>
            <Button text="Загрузить" onAction="#handleLoadGame"/>
            <Button text="Сохранить" onAction="#handleSaveGame"/>
            <Button text="Настройки" onAction="#handleSettings"/>
        </HBox>
    </top>

    <left>
        <VBox spacing="5" padding="10">
            <Label text="Игрок: Player1"/>
            <Label text="Уровень: 5"/>
            <Label text="Золото: 1000"/>
        </VBox>
    </left>

    <center>
        <GridPane fx:id="gameBoard" gridLinesVisible="true"/>
    </center>

    <right>
        <VBox spacing="10" padding="10" prefWidth="200">
            <Label text="Выберите юнита" wrapText="true"/>
            <Button text="Двигаться" onAction="#handleMove" maxWidth="Infinity"/>
```

Контроллер для FXML

```
public class GameController {
    @FXML private GridPane gameBoard;
    @FXML private Label unitInfo;

    private GameEngine gameEngine;
    private Unit selectedUnit;

    @FXML
    public void initialize() {
        gameEngine = new GameEngine();
        setupGameBoard();
        updateDisplay();
    }

    @FXML
    private void handleNewGame() {
        gameEngine.startNewGame();
        updateDisplay();
    }

    @FXML
    private void handleLoadGame() {
        FileChooser fileChooser = new FileChooser();
        fileChooser.setTitle("Загрузить игру");
        fileChooser.getExtensionFilters().add(
            new FileChooser.ExtensionFilter("Файлы сохранений", "*.save", "*.json")
        );

        File file = fileChooser.showOpenDialog(gameBoard.getScene().getWindow());
        if (file != null) {
            gameEngine.loadGame(file);
            updateDisplay();
        }
    }

    @FXML
    private void handleMove() {
        if (selectedUnit != null) {
            // Логика движения
        }
    }

    @FXML
    private void handleAttack() {
        if (selectedUnit != null) {
            // Логика атаки
        }
    }
}
```

Загрузка FXML

```
public class GameApplication extends Application {

    @Override
    public void start(Stage primaryStage) {
        try {
            // Загрузка FXML
            FXMLLoader loader = new FXMLLoader(getClass().getResource("game_interface.fxml"));
            Parent root = loader.load();

            // Получение контроллера
            GameController controller = loader.getController();

            // Создание сцены
            Scene scene = new Scene(root);

            // Настройка окна
            primaryStage.setTitle("Пошаговая стратегия");
            primaryStage.setScene(scene);
            primaryStage.show();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Анимации и эффекты

Простые анимации:

```
// Анимация движения юнита
private void animateUnitMove(Unit unit, Position from, Position to) {
    // Создание анимации
    TranslateTransition moveAnimation = new TranslateTransition(Duration.millis(500));
    moveAnimation.setNode(getUnitNode(unit));
    moveAnimation.setFromX(from.getX() * 60);
    moveAnimation.setFromY(from.getY() * 60);
    moveAnimation.setToX(to.getX() * 60);
    moveAnimation.setToY(to.getY() * 60);

    // Запуск анимации
    moveAnimation.play();
}

// Эффект выделения
private void highlightUnit(Unit unit) {
    Node unitNode = getUnitNode(unit);

    // Эффект свечения
    Glow glow = new Glow();
    glow.setLevel(0.8);
    unitNode.setEffect(glow);

    // Анимация пульсации
    ScaleTransition pulse = new ScaleTransition(Duration.millis(1000), unitNode);
    pulse.setFromX(1.0);
    pulse.setToX(1.1);
    pulse.setFromY(1.0);
```

Стилизация с CSS

game_styles.css:

```
.root {
    -fx-background-color: #2b2b2b;
    -fx-text-fill: white;
}

.button {
    -fx-background-color: #4a4a4a;
    -fx-text-fill: white;
    -fx-border-color: #6a6a6a;
    -fx-border-radius: 3;
    -fx-background-radius: 3;
    -fx-padding: 8 16;
}

.button:hover {
    -fx-background-color: #5a5a5a;
    -fx-border-color: #7a7a7a;
}

.button:pressed {
    -fx-background-color: #3a3a3a;
}

.label {
    -fx-text-fill: #e0e0e0;
    -fx-font-size: 14px;
}

.game-board {
    -fx-background-color: #1e1e1e;
    -fx-border-color: #4a4a4a;
    -fx-border-width: 2;
}

.cell {
    -fx-fill: #3a3a3a;
    -fx-stroke: #5a5a5a;
    -fx-stroke-width: 1;
}

.cell:hover {
    -fx-fill: #4a4a4a;
    -fx-stroke: #6a6a6a;
}

.unit-warrior {
    -fx-fill: #8b4513;
}
```

Применение CSS

```
// Загрузка CSS файла
scene.getStylesheets().add(getClass().getResource("game_styles.css").toExternalForm());

// Применение стилей к элементам
unitNode.getStyleClass().add("unit-" + unit.getType().toString().toLowerCase());

// Динамическое изменение стилей
unitNode.getStyleClass().add("selected");

// Удаление стилей
unitNode.getStyleClass().remove("selected");
```

Диалоговые окна

```
public class DialogManager {

    public static void showGameOverDialog(String winner) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Игра окончена");
        alert.setHeaderText("Победитель: " + winner);
        alert.setContentText("Хотите начать новую игру?");

        ButtonType newGameButton = new ButtonType("Новая игра");
        ButtonType exitButton = new ButtonType("Выход");
        alert.getButtonTypes().setAll(newGameButton, exitButton);

        Optional<ButtonType> result = alert.showAndWait();
        if (result.isPresent() && result.get() == newGameButton) {
            // Начать новую игру
        } else {
            // Выйти из игры
            Platform.exit();
        }
    }

    public static String showSaveGameDialog() {
        TextInputDialog dialog = new TextInputDialog();
        dialog.setTitle("Сохранение игры");
        dialog.setHeaderText("Введите название сохранения:");
        dialog.setContentText("Название:");

        Optional<String> result = dialog.showAndWait();
        return result.orElse(null);
    }
}
```

Практический пример: Полный игровой интерфейс

```
public class GameInterface {
    private Stage primaryStage;
    private GameEngine gameEngine;
    private GameController controller;

    public void start() {
        // Создание главного окна
        primaryStage = new Stage();
        primaryStage.setTitle("Пошаговая стратегия");
        primaryStage.setWidth(1400);
        primaryStage.setHeight(900);
        primaryStage.setMinWidth(1000);
        primaryStage.setMinHeight(700);

        try {
            // Загрузка интерфейса
            FXMLLoader loader = new FXMLLoader(getClass().getResource("game_interface.fxml"));
            Parent root = loader.load();

            // Получение контроллера
            controller = loader.getController();
            controller.setGameEngine(gameEngine);

            // Создание сцены
            Scene scene = new Scene(root);
            scene.getStylesheets().add(getClass().getResource("game_styles.css").toExternalForm());

            // Настройка горячих клавиш
            setupHotkeys(scene);

            // Установка сцены
            primaryStage.setScene(scene);
            primaryStage.show();

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void setupHotkeys(Scene scene) {
        scene.setOnKeyPressed(e -> {
            switch (e.getCode()) {
                case ESCAPE: controller.handleEscape(); break;
                case ENTER: controller.handleEnter(); break;
```

Лучшие практики JavaFX

✅ Что делать:

- **Разделять логику и интерфейс** с помощью FXML
- **Использовать CSS** для стилизации
- **Обрабатывать события** асинхронно
- **Применять анимации** для улучшения UX
- **Тестировать** на разных разрешениях

❌ Чего избегать:

- **Смешивать логику** с интерфейсом
- **Блокировать UI поток** длительными операциями
- **Игнорировать** доступность

Домашнее задание

Задача 1:

Создать базовый интерфейс игры с использованием FXML

Задача 2:

Реализовать обработку событий мыши и клавиатуры

Задача 3:

Добавить анимации для движения юнитов

Что дальше?

На следующей лекции:

- **Паттерны проектирования**
- **Singleton, Factory**
- **Observer, Strategy**
- **Применение в игре**

Подготовка:

- Изучить главу 15-16 из учебника
- Выполнить домашнее задание
- Подготовить вопросы по текущей теме

Вопросы?

Контакты:

- **Email:** [ваш.email@university.edu]
- **Telegram:** [@username]
- **Офис:** [номер кабинета]

Следующая лекция: Паттерны проектирования