Урок 6

Управление «экранами»

Разделение игры на экраны. Менеджер экранов. Меню, настройки, таблица результатов. Пользовательский интерфейс. Переходы между экранами, освобождение ресурсов

Управляем экранами

В игре может быть несколько экранов: меню, опций, таблицы результатов, самой игры и т.д. Прописывать логику этих элементов в одном классе очень неудобно, поэтому, как правило, создается несколько классов, каждый из которых отвечает за обработку своего экрана. В LibGDX есть готовый интерфейс Screen, описывающий работу экранов.

```
public class MyScreen implements Screen {
    @Override
    public void show() {
    }

    @Override
    public void render(float delta) {
    }

    @Override
    public void resize(int width, int height) {
    }

    @Override
    public void pause() {
    }

    @Override
    public void resume() {
    }

    @Override
    public void hide() {
    }

    @Override
    public void dispose() {
    }
}
```

Выше представлен код класса, реализующего экранный интерфейс. Каждый из методов выполняет определенную задачу: **show()** выполняется при переключении на данный экран и отвечает за подготовку экрана к работе, **render()** отвечает за отрисовку графики, **resize()** обработка изменения размера окна, **pause()** срабатывает когда экран приостанавливается (например, при сворачивании приложения), **resume()** отвечает за обработку возвращения программы из паузы, **hide()** выполняется при переключении на другой экран, **dispose()** вызывается когда экран должен освободить занятые ресурсы.

В нашей игре мы сделали два экрана: основной игровой (GameScreen) и меню (MenuScreen), которые реализуют интерфейс Screen. Для того чтобы иметь возможность переключаться между экранами,

необходимо изменить основной класс игры MyGdxGame, если раньше он наследовался от класса ApplicationAdapter, то теперь необходимо его унаследовать от Game (в классе Game присутствует метод setScreen()). Чтобы иметь возможность из любого экрана переключаться на любой другой экран, желательно создать управляющий синглтон класс ScreenManager, код которого приведен ниже.

```
public class ScreenManager {
    enum ScreenType {
        MENU, GAME
    }

    private static final ScreenManager ourInstance = new ScreenManager();

    public static ScreenManager getInstance() {
        return ourInstance;
    }

    private Game game;

    private MenuScreen menuScreen;
    private GameScreen gameScreen;

    public void init(Game game) {
        this.game = game;
        this.menuScreen = new MenuScreen(((StarGame)game).batch);
        this.gameScreen = new GameScreen(((StarGame)game).batch);
    }
    public void switchScreen(ScreenType type) {
        Screen screen = game.getScreen();
        Assets.getInstance().assetManager.clear();
        Assets.getInstance().assetManager.dispose();
        Assets.getInstance().assetManager = new AssetManager();
        if (screen != null) {
            screen.dispose();
        }
        switch (type) {
            case MENU:
                Assets.getInstance().loadAssets(ScreenType.MENU);
                game.setScreen(menuScreen);
                break;
            case GAME:
                Assets.getInstance().loadAssets(ScreenType.GAME);
                game.setScreen(gameScreen);
                break;
        }
    }
    private ScreenManager() {
    }
    public void dispose() {
    }
}
```

ScreenManager хранит в себе перечисление (enum) возможных экранов (ScreenType.MENU и ScreenType.GAME), и ссылку на основной игровой класс Game game. При запуске игры MyGdxGame.create() мы инициализируем менеджер экранов через метод init(), который запоминает ссылку на игру и подготавливает к работе экраны. Для того чтобы переключиться на один из экранов, необходимо обратиться к ScreenManager и вызвать у него метод switchScreen() с типом нужного экрана, после чего game переключит экран на указанный и освободит ресурсы предыдущего экрана через класс Assets.

После появления нескольких экранов подверглась изменению структура класса, управляющего ресурсами, поскольку для разных экранов нужны разные ресурсы.

```
public class Assets {
    private static final Assets ourInstance = new Assets();

    public static Assets getInstance() {
        return ourInstance;
    }

    AssetManager assetManager;

    TextureAtlas mainAtlas;

    private Assets() {
        assetManager = new AssetManager();
    }

    public void loadAssets(ScreenManager.ScreenType type) {
        switch (type) {
            case MENU:
                assetManager.load("my.pack", TextureAtlas.class);
                assetManager.load("bg.png", Texture.class);
                assetManager.finishLoading();
                mainAtlas = assetManager.get("my.pack", TextureAtlas.class);
                break;
            case GAME:
                assetManager.load("font.fnt", BitmapFont.class);
                assetManager.load("my.pack", TextureAtlas.class);
                assetManager.load("bg.png", Texture.class);
                assetManager.load("powerUps.png", Texture.class);
                assetManager.finishLoading();
                mainAtlas = assetManager.get("my.pack", TextureAtlas.class);
                break;
        }
    }
}
```

При загрузке какого-либо из экранов достаточно обратиться к синглтон классу Assets и вызвать у него метод loadAssets() с указанием типа экрана, после чего Assets сам загрузит необходимые для этого экрана ресурсы. Для небольшого упрощения кода Assets хранит ссылку на основной атлас текстур mainAtlas.

После добавления и корректировки классов метод create() класса StarGame выглядит следующим образом:

```
@Override
public void create() {
    batch = new SpriteBatch();
    ScreenManager.getInstance().init(this);
    ScreenManager.getInstance().switchScreen(ScreenManager.ScreenType.MENU);
}
```

Создаем batch, инициализируем менеджер экранов и переключаемся на экран меню. Запуск основного игрового экрана тоже подвергся небольшим изменения:

```
@Override
public void show() {
    fnt = Assets.getInstance().assetManager.get("font.fnt", BitmapFont.class);
    fireBtnRegion = Assets.getInstance().mainAtlas.findRegion("btExit");
    background = new Background();
    powerUps = new ArrayList<PowerUp>();
    hero = new Hero();
    EnemiesEmitter.getInstance().reset();
    BulletEmitter.getInstance().reset();
    for (int i = 0; i < 3; i++) {
        EnemiesEmitter.getInstance().setupAsteroid((float) Math.random() * 1280, (float) Math.random() * 720,
1.0f, 100);
    }
    for (int i = 0; i < 1; i++) {
        EnemiesEmitter.getInstance().setupBot(hero, (float) (1280 * Math.random()), (float) (720 *
Math.random()), 50, 20);
    }
}
```

При каждом переходе на игровой экран (даже если мы вышли в меню и потом опять начали новую игру) игровые ресурсы загружаются снова, игрок пересоздается, а имеющиеся до этого боты, астероиды и пули уничтожаются, чтобы при повторных запусках игры не было непонятных старых объектов на экране.

Домашнее задание

1. На странице занятия

Дополнительные материалы

1. https://github.com/libgdx/libgdx/wiki/Extending-the-simple-game