УО «ВГУ им. П.М. Машерова»
Кафедра прикладной математики и механики

С. А. Ермоченко

Технологии Adobe® Flash® и платформа Adobe® AIR®

Материалы для практических занятий по дисциплине «Дополнительные главы информатики. Современные технологии программирования» для студентов 3 курса специальности Прикладная математика (1-31 03 03)

Витебск, 2011

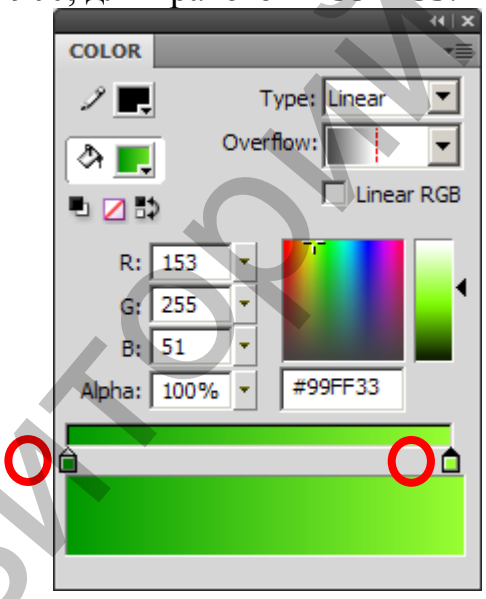
Лабораторная работа №1
Создание анимированного Flash-ролика

Задание 1

Создать простейший ролик с прыгающими мячами.

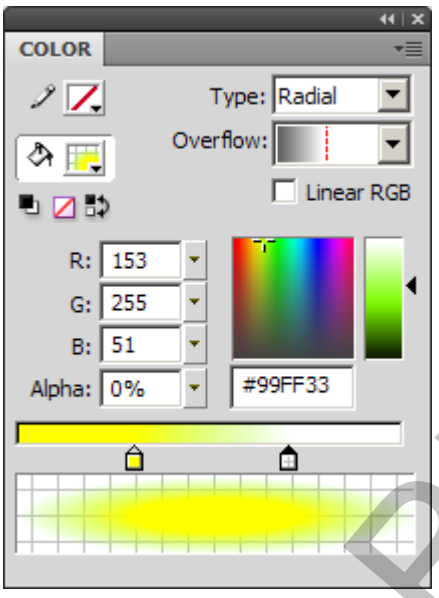
Ход выполнения работы:

1. Создайте новый файл «Flash File (Action Script 3.0)». Установите в панели «Properties» размер рабочей области «Size» 800 на 600 пикселей, цвет фона «Stage» – #99CCFF. Переименуйте слой «Layer 1» в «Фон».
2. Откройте панель «Color» (Window > Color), выберете в качестве типа заливки «Type» линейный градиент «Linear». Установите для левого ползунка цвет #009900, для правого – #99FF33:

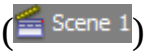


Выберите инструмент «Rectangle Tool» панели «Tools». Задайте для него в панели «Properties» отсутствие обводки ( ). Нарисуйте прямоугольник и выделите его инструментом «Selection Tool». Поверните его на 90° против часовой стрелки (Modify > Transform > Rotate 90° CCW). На панели «Properties» задайте ширину прямоугольника 800 пикселей, высоту – 100, координату *x* левого верхнего угла – 0, координату *y* – 500.

3. Выберите инструмент «Oval Tool» (находится «за» инструментом «Rectangle Tool»). Нарисуйте окружность диаметром 120 пикселей в правом верхнем углу рабочей области. На панели «Color» смените тип градиента на «Radial», цвет градиента #009900 измените на #FFFF00, а цвет #99FF33 сделайте прозрачным (Параметр «Alpha» 0%).



4. Заблокируйте слой «Фон».
5. Создайте новый символ и назовите его «Красный мяч» (Insert > New Symbol…). Нарисуйте круг диаметром 150 пикселей в центре символа (x = -75; y = -75) с радиальной градиентной заливкой (внешний цвет #FF0000, внутренний – #FF9966) и черным внешним контуром (будьте внимательны, при этом заливка и контур будут созданы как две отдельные фигуры). **Если необходимо выделить несколько фигур одновременно, они выделяются с зажатой клавишей Shift.**

   Создайте еще одну окружность диаметром 250 пикселей и расположением x = -50; y = -200 без заливки с черным контуром. Две созданные окружности пересекутся, при этом автоматически произойдет разрезание фигур на непересекающиеся фигуры. Удалите верхнюю часть большой окружности. Создайте еще одну окружность без заливки с черным контуром, диаметром 250 пикселей и с расположением x = -60; y = -190. Также после пересечения фигур (пересечение происходит **после снятия выделения**) удалите верхнюю часть большой окружности. Выделите нижнюю часть мяча и смените цвета градиента для нее: внешний – #CC0000, внутренний – #CC6633. Для полоски на мяче задайте цвета градиента внешний – #FF9900, внутренний – #FFFF00.
6. Вернитесь на сцену с помощью вкладки «Scene 1» (

). Создайте новый слой с именем «Красный мяч». Из панели «Library» перетащите мышью символ «Красный мяч». Установите его в позицию (-75; 150). На обоих созданных слоях вставьте новый фрейм в 75-ом кадре (Команда «Insert Frame» из контекстного меню кадра). Создайте анимацию для мяча (Команда «Create Motion Tween» из контекстного меню символа). Помесите курсор на 25-ый кадр в панели «Timeline». После чего переместите мяч в позицию (250; 425). Аналогично измените его позицию в 50-ом кадре на (550; 425), в 70-ом – (800; 425),

в 75-ом – (875; 350). Далее инструментом «Selection Tool» отредактируйте кривую пути мяча, придав каждому отрезку вид дуги. Просмотрите полученную анимацию, если необходимо, подкорректируйте путь мяча для придания ему более естественного движения.

7. Для имитации упругости мяча в 25-ом кадре уменьшите его высоту до 130 пикселей, а позицию по вертикали увеличьте до 435. В 50-ом кадре соответствующие параметры установите в 140 и 430. Выберите между 25-м и 50-м кадром кадр, соответствующий наивысшей точке мяча, для этого кадра задайте высоту мяча 170 пикселей. Аналогично между 50-м и 70-м кадром в наивысшей точке установите высоту мяча 160 пикселей. При просмотре анимации можно заметить, что между 1-ым и 25-ым кадром высота мяча плавно уменьшается, что неестественно до 1-го удара о землю. Для устранения этого в 24-ом кадре задайте высоту мяча стандартному для него значению в 150 пикселей.
8. Добавим вращение мяча вокруг своей оси. Для этого необходимо добавить анимацию внутри самого символа. Каждый символ имеет свою шкалу времени, анимация на которой проигрывается параллельно с основной. Для создания этой анимации необходимо на панели «Library» дважды кликнуть по изображению символа для открытия окна его редактирования. После чего нужно выделить те части символа, которые будут анимированы (в нашем случае выбираем все). В контекстном меню группы выделенных объектов выбираем пункт «Create Motion Tween», после чего будет предложено создать еще один «символ в символе». Далее анимация настраивается как обычно. В нашем случае создадим фрейм в 60-ом кадре символа, после чего выделим первый кадр и в панели «Properties» для параметра «Rotate» зададим значение «1 time(s)».
9. Создайте новый слой «Зеленый мяч». Скопируйте в него кадры с 1-го по 75-ый со слоя «Красный мяч» в кадры с 76-го по 150-ый. В 76-ом кадре слоя «Зеленый мяч» в контекстном меню символа с мячом выберите пункт «Duplicate Symbol…» и для нового символа задайте имя, измените направление вращения мяча (на панели «Properties» в выпадающем меню «Direction» вместо «CW» выбрать «CCW»). Затем в окне редактирования символа еще раз выберите пункт «Duplicate Symbol…» из контекстного меню и создайте копию символа, в которой поменяйте цвет мяча (можно просто менять местами красные и зеленые компоненты). После этого вернитесь на основную сцену и выделите путь движения зеленого мяча и отразите его слева направо (Modify > Transform > Flip Horizontal). Уменьшите размеры зеленого мяча до 100 пикселей.
10. Создайте еще один символ с летающим насекомым и анимируйте его для имитации полета. Для этого удобно нарисовать крылья в виде овалов и в нескольких фреймах немного поворачивать их вокруг точки, в которой крыло крепиться к насекомому. Для этого удобно

воспользоваться инструментом «Free Transform Tool», при этом вокруг любого выделенного объекта отображается специальный прямоугольник. Поворот объекта осуществляется подведением указателя мыши к одному из углов прямоугольника (возле указателя должна появиться круглая стрелка). После чего можно вращать фигуру вокруг центра вращения. По-умолчанию цент вращения помещается в центре прямоугольника (специальная белая круглая точка). Но ее можно перетащить мышью в любую нужную точку. После создания символа создайте на главной сцене анимацию с использованием созданного символа вдоль некоторого пути. В настройках пути движения на панели «Properties» отметьте галочкой пункт «Orient to path».

Задание 2

Создайте анимацию некоторого объекта, движущегося по некоторой траектории. Объект должен появляться на сцене, проходить по некоторому пути, и исчезать со сцены. Затем опять появляться, проходить по новому пути, и исчезать. Различных путей должно быть как минимум **четыре**. В качестве объектов можно использовать: различный транспорт (авто-, водо-, аэро-…); спортивные объекты (шары для боулинга, бильярда…) и т.д.

Лабораторная работа №2
Управление анимацией в ActionScript

Задание 1

Добавить элементы случайности в ролик с прыгающими мячами.

Ход выполнения работы:

1. По аналогии с предыдущей лабораторной добавьте в ролик еще два мяча (например, синий и желтый).
2. Создайте новый слой, назовите его «метки». Вставьте ключевые фреймы в слое «метки» в кадры, соответствующие началу анимации каждого мяча.
3. Каждому ключевому фрейму слоя «метки» присвойте соответствующую метку (в панели «Properties» для соответствующего кадра в разделе «Label» присвоить имя в поле «Name»).
4. Создайте новый слой, назовите его «действия». Вставьте ключевые фреймы в слое «действия» в кадры, соответствующие окончанию анимации каждого мяча.
5. Для каждого ключевого фрейма слоя «действия», соответствующего окончанию анимации мяча, вставьте код на языке ActionScript (из контекстного меню кадра пункт «Actions»).
6. В соответствующем коде необходимо сгенерировать случайным образом целое число от 0 до 3 (каждое значение для мяча определенного цвета), а затем осуществить переход на метку, которой помечена анимация мяча, соответствующего сгенерированному числу.

Примечания:

Для генерации числа нужно использовать статический метод класса `Math`:

```
public static function random() : Number
```

который возвращает случайное дробное число в диапазоне от 0 (включительно) до 1 (не включая).

Для преобразования дробного числа в целое можно воспользоваться статическим методом класса `Math`:

```
public static function floor(val : Number) : Number
```

который возвращает ближайшее целое число, не превосходящее данное число `val`.

Для перехода на определенную метку на шкале времени используется метод класса `MovieClip`:

```
public function gotoAndPlay(frame : Object,
                            scene : String = null) : void
```

где:

`frame` может быть строкой, содержащей метку, которой помечен необходимый кадр;

`scene` – строка, содержащая имя сцены, на которой располагается необходимый кадр. Если данный параметр равен `null`, кадр будет искаться на текущей сцене.

Объект класса `MovieClip` ставиться в соответствие каждому ролику, созданному во Flash. Любой код ActionScript, выполняющийся в кадре ролика, фактически выполняется внутри некоторого метода соответствующего объекта класса `MovieClip`. Это означает, что для вызова любого из методов класса `MovieClip` внутри фрейма некоторого ролика, достаточно воспользоваться ссылкой `this`, указывающей на соответствующий объект класса `MovieClip`.

Задание 2

Аналогичным образом добавьте элемент случайности в ролик задания 2 предыдущей лабораторной работы. Введите в ролик некоторый числовой параметр, который будет менять свое значение в зависимости от анимации. Реализуйте динамическое отображение этого параметра в ролике.

Лабораторная работа №3
Создание простейшего Flash-приложения

Задание 1

Разработать программу, рассчитывающую полет тела, брошенного под углом к горизонту.

Ход выполнения работы:

1. Создать ролик с размером сцены 650 на 400 пикселей и цветом фона #99AABB
2. Переименовать существующий слой в «Форма».
3. С помощью команды меню «Window > Components» (Ctrl+F7) открыть панель «Components».
4. Разместить на слое два компонента «Label» с шириной 140 пикселей, высотой – 20, координатой *x* левого верхнего угла – 5, а координату *y* задать соответственно равной 5 и 30 пикселям (перечисленные свойства задаются в панели «Properties»).
5. С помощью команды меню «Window > Component Inspector» (Shift+F7) открыть панель «Component Inspector».
6. В параметрах каждой метки установить значение параметра «autoSize» равным «right», а параметр «text» для меток с верху вниз установить в «Угол броска (°):» и «Начальная скорость (м/с):» соответственно.
7. Разместить на том же слое два компонента «TextInput» с шириной 100 пикселей, высотой и координатой *y* верхнего левого угла как у меток, а координатой *x* равной 150 пикселей. Полям в порядке сверху вниз присвоить имена соответственно «fieldAngle» и «fieldSpeed».
8. Разместить на слое «Форма» компонент «Button» с размерами 100 на 20 пикселей, в позиции 150; 55. Присвоить ему имя «buttonGo». Параметру «label» присвоить значение «Пуск».
9. Разместить на слое «Форма» два компонента «Label» с шириной 390, высотой 20, координатой *x* левого верхнего угла 255, координатой *y* левого верхнего угла 5 и 30 пикселей соответственно. Параметру «text» присвоить пустую строку, дать меткам имена «errorAngle» и «errorSpeed» соответственно.
10. Создайте новый слой «Действия». В первом фрейме этого слоя разместите следующий код на языке ActionScript:

```
buttonGo.addEventListener(MouseEvent.CLICK,
                          buttonGoClick);
stop();

function buttonGoClick(e:MouseEvent):void {
}
```

11. Создайте символ – шар диаметром в 10 пикселей, центрируйте его относительно символа. Экспортируйте его для ActionScript под именем «Stone».
12. Создайте новый слой «Метки». В первом, втором и третьем кадре слоя создайте ключевые фреймы. Второму из них присвойте метку «beginAnimation», третьему – «endAnimation».
13. В первом фрейме слоя «Действия» добавьте в функцию `buttonClick` строчку

```
gotoAndPlay("beginAnimation");
```

14. Вставьте во втором, третьем и четвертом кадрах слоя «Действия» вставьте ключевые кадры. В первом из них вставьте код ActionScript^

```
var stone:Stone = new Stone();
stone.y = stage.stageHeight;
addChild(stone);
```

15. В третьем фрейме добавьте код ActionScript:

```
stone.x++;
stone.y--;
```

16. В четвертом фрейме добавьте код ActionScript:

```
gotoAndPlay("endAnimation");
```

17. Просмотрите полученный ролик. При старте Вы должны увидеть форму с кнопкой. При нажатии на кнопку форма должна пропасть, а появится шарик, летящий из нижнего левого угла.
18. Добавим теперь обработку входных данных. Сначала необходимо произвести проверку корректности данных. Для этого модифицируем код первого фрейма слоя «Действия» следующим образом:

```
var angle:Number;
var speed:Number;
buttonGo.addEventListener(MouseEvent.CLICK, buttonGoClick);
stop();

function buttonGoClick(e:MouseEvent):void {
    errorAngle.text = new String();
    errorSpeed.text = new String();
    if(fieldAngle.text.length == 0) {
```

```
            errorAngle.text = "Введите значение угла броска";
            return;
        }
        angle = new Number(fieldAngle.text);
        if(isNaN(angle)) {
            errorAngle.text =
                    "Значение угла броска должно быть числовым";
            return;
        }
        if(angle <= 0 || angle >= 90) {
            errorAngle.text =
                  "Угол броска должено быть в интервале (0; 90)";
            return;
        }
        if(fieldSpeed.text.length == 0) {
            errorSpeed.text =
                          "Введите значение начальной скорости";
            return;
        }
        speed = new Number(fieldSpeed.text);
        if(isNaN(speed)) {
            errorSpeed.text =
              "Значение начальной скорости должно быть числовым";
            return;
        }
        if(speed <= 0 || speed > 100) {
            errorSpeed.text =
           "Начальная скорость должна быть в интервале (0; 100]";
            return;
        }
        gotoAndPlay("beginAnimation");
    }
```

19. Во втором фрейме слоя «Действия» добавьте в конце следующий код:

```
var time:Number = 0;
var phi:Number = Math.PI * angle / 180;
var sp:Number = Math.sin(phi);
var cp:Number = Math.cos(phi);
var g:Number = 9.81;
var flyingTime:Number = 2 * speed * sp / g;
var flyingLength:Number = 2 * speed * speed * sp * cp / g;
var flyingHeight:Number = speed * speed * sp * sp / (2 * g);
var len:Number;
```

```
if(flyingLength * stage.stageHeight >
                              flyingHeight * stage.stageWidth) {
    len = stage.stageWidth / flyingLength;
} else {
    len = stage.stageHeight / flyingHeight;
}
var a:Number;
var b:Number;
```

20. Код третьего фрейма замените на:

```
a = speed * time * cp;
b = speed * time * sp - g * time * time / 2;
time += 0.04;
stone.x = a * len;
stone.y = stage.stageHeight - b * len;
```

21. В первом кадре слоя «Метки» добавите метку «formLabel». В четвертом фрейме слоя «Действия» код замените на:

```
if(time <= flyingTime) {
    gotoAndPlay("endAnimation");
} else {
    removeChild(stone);
    gotoAndPlay("formLabel");
}
```

22. Убедитесь в работоспособности ролика. Добавьте возможность вывода в некоторой фиксированной области ролика координаты летящего тела в каждый момент времени (переменные `a` и `b`). Рядом с телом выводите его текущую скорость. Компоненты этой скорости по осям: (`speed * cp`; `speed * sp – g * time`).

Задание 2

Вариант 1.

Дан бассейн некоторого объема (задается пользователем). В бассейне есть источник и сток воды. Объем воды, поступающий в бассейн в единицу времени, задается функцией $f(t) = a\,[1 - \cos(b\,t)]$, объем воды, вытекающий из бассейна, задается функцией:

$$g(t)=\begin{cases} c\ t-T\ ^2,\ \text{ïðè}\ \ 0;\ 2T \\ c\left[2T^2-\ t-3T\ ^2\right],\ \text{ïðè}\ \ 2T;\ 4T \\ g\ t-4T\ ,\ \text{ïðè}\ \ 4T;\ \infty \\ g\ t+4T\ ,\ \text{ïðè}\ \ -\infty;\ 0 \end{cases}$$

Создать анимацию, моделирующую процесс наполнения бассейна водой в течение времени. При переполнении бассейна выдать соответствующее сообщение и закончить анимацию, дав возможность пользователю ввести новые данные. Параметры функций (*a*, *b* и *c*) и период *T* функции *g*(*t*) вводятся пользователем.

Вариант 2.

В некотором проводнике напряжение меняется с течением времени по закону:

$$U\ t\ =a\cos\ bt\ +\frac{c}{t}$$

Сила тока меняется по закону:

$$I(t)=\begin{cases} d\left[\text{ch}\ t-T\ -\text{ch}\ T\ \right],\ \text{ïðè}\ \ 0;\ 2T \\ d\left[\text{ch}\ T\ -\text{ch}\ t-3T\ \right],\ \text{ïðè}\ \ 2T;\ 4T \\ I\ t-4T\ ,\ \text{ïðè}\ \ 4T;\ \infty \\ I\ t+4T\ ,\ \text{ïðè}\ \ -\infty;\ 0 \end{cases}$$

Параметры *a*, *b*, *c* и *d*, а также период *T* функции *I*(*t*) задаются пользователем. $\text{ch}\ x\ =\left[e^x+e^{-x}\right]/2$ – гиперболический косинус.

Необходимо создать анимированный омметр, измеряющий сопротивление проводника. Сопротивление есть произведение напряжения на силу тока. Омметр выполнить как прибор со стрелкой, отклоняющейся на необходимый угол. Минимальное и максимальное значения шкалы автоматически рассчитать по введенным пользователем параметрам.

Вариант 3.

Создать анимацию авиа-радара, отображающего пролетающие в радиусе действия самолеты. Центр радара расположить в центре ролика. Радар должен высвечивать точку–летательный аппарат в диапазоне 30 градусов луча радара (то есть если точка находится непосредственно на луче радара, она отображается с максимальной яркостью, если она не находится в 30-градусном шлейфе от луча радара, она не отображается, в

пределах шлейфа ее яркость линейно зависит от близости к лучу рада).

Точка движется по экрану радара по одной из траекторий, каждый раз случайно выбираемых, которые заданы своими уравнениями в системе координат, связанной с центром радара, в случайном направлении:

$$y = a\ln(bx),$$

$$y = a\,\mathrm{arctg}(bx+1),$$

$$y = e^{-ax} - bx,$$

$$y = ax^3 - b,$$

Число оборотов луча радара задаются пользователем. Параметры *a* и *b* случайно выбираются из вводимого пользователем диапазона.

Вариант 4.

Дан аквариум с заданной пользователем глубиной, длиной и шириной. Аквариум отображается прямоугольником с заданными глубиной и длиной (ширина нужна для вычисления его объема). Аквариум на 2/3 заполнен водой. Пользователь вводит массу и плотность шарика, бросаемого в воду. Необходимо анимировать погружение этого шарика в воду с учетом сопротивления воды (сопротивление воздуха не учитывать). Место погружения каждый раз выбирать случайным образом.

Во время полета шарика от верхней кромки аквариума (в этой точке начальная скорость равна 0) до верхней кромки воды на шарик действует только сила тяжести $F_T = m\,g$ (g = 9,81 м/с^2). После погружения в воду на шарик начинает действовать еще и Архимедова сила, выталкивающая шарик из воды. Численно она равна весу жидкости в объеме, равном объему шарика. Зная суммарную силу F, действующую на шарик в каждый момент времени, и массу шарика m можно найти его ускорение a из соотношения $F = m\,a$. Зная ускорение, можно интегрированием найти скорость и перемещение.

Вариант 5.

Создать анимацию, демонстрирующую рост дерева. Само дерево рисуется отрезками, строящимися по определенному правилу. Рост каждой ветки определенным образом зависит от родительской ветки. При чем такая зависимость для веток каждого уровня одинакова. Для каждой родительской ветки задается три дочерние ветки, при чем для каждой дочерней ветки задается точка, из которой она будет расти на родительской ветке. Эта точка задается процентным соотношением (0% –

дочерняя ветка растет из той же точки, что и родительская, 100% – дочерняя ветка растет из вершины родительской). Также для каждой ветви задается процентное соотношение ее длины к длине родительской ветви. Кроме этого для каждой ветви задается угол ее наклона к родительской (знак угла определяет направление отсчета). Задается общее число уровней, при чем ветви последнего уровня должны быть листьями.

Таким образом, достаточно изменять высоту ствола дерева, остальные ветви пересчитываются рекурсивно.

Лабораторная работа №4
Обработка данных во Flash-приложении

Задание 1

На основе приведенных ниже фрагментов кода создать Flash-приложения для управления списком продуктов. Добавить в него возможность автоматического подсчета общей стоимости всех продуктов в списке (при возникновении события `DataGridEvent.ITEM_EDIT_END`).

Листинг 1. – класс Product

```
package data {
    public class Product {
        private var identityProp : int;
        private var categoryProp : String;
        private var nameProp : String;
        private var amountProp : int;
        private var priceProp : Number = 0;
        public function get identity() : int {
            return identityProp;
        }
        public function set identity(identityProp : int) :
                                                    void {
            if(identityProp > 0) {
                this.identityProp = identityProp;
            }
        }
        public function get category() : String {
            return categoryProp;
        }
        public function set category(categoryProp : String) :
                                                    void {
            if(categoryProp != null &&
                                categoryProp.length > 0) {
                this.categoryProp = categoryProp;
            }
        }
        public function get name() : String {
            return nameProp;
        }
        public function set name(nameProp : String) : void {
            if(nameProp != null && nameProp.length > 0) {
                this.nameProp = nameProp;
```

```
            }
        }
        public function get amount() : int {
            return amountProp;
        }
        public function set amount(amountProp : int) : void {
            if(amountProp >= 0) {
                this.amountProp = amountProp;
            }
        }
        public function get price() : Number {
            return priceProp;
        }
        public function set price(priceProp : Number) : void{
            if(!isNaN(priceProp) && priceProp >= 0) {
                this.priceProp = priceProp;
            }
        }
    }
}
```

Листинг 2. – класс Table

```
package view {
    import fl.controls.DataGrid;
    import fl.data.DataProvider;
    import flash.display.Sprite;
    import fl.controls.dataGridClasses.DataGridColumn;
    import fl.events.DataGridEvent;

    public class Table extends Sprite {
        private var dataGrid : DataGrid;

        public function Table() {
            dataGrid = new DataGrid();
            dataGrid.x = 5;
            dataGrid.y = 5;
            dataGrid.width = 540;
            dataGrid.height = 365;
            dataGrid.allowMultipleSelection = true;
            dataGrid.editable = true;
            var identityColumn : DataGridColumn =
                                        new DataGridColumn();
            identityColumn.dataField = "identity";
            identityColumn.headerText = "№";
```

```
            identityColumn.width = 40;
            identityColumn.editable = false;
            var categoryColumn : DataGridColumn =
                                        new DataGridColumn();
            categoryColumn.dataField = "category";
            categoryColumn.headerText = "Категория";
            categoryColumn.width = 100;
            var nameColumn : DataGridColumn =
                                        new DataGridColumn();
            nameColumn.dataField = "name";
            nameColumn.headerText = "Наименование";
            var amountColumn : DataGridColumn =
                                        new DataGridColumn();
            amountColumn.dataField = "amount";
            amountColumn.headerText = "Количество";
            amountColumn.width = 80;
            var priceColumn : DataGridColumn =
                                        new DataGridColumn();
            priceColumn.dataField = "price";
            priceColumn.headerText = "Цена";
            priceColumn.width = 80;
            dataGrid.columns = [
                identityColumn,
                categoryColumn,
                nameColumn,
                amountColumn,
                priceColumn
            ];
            addChild(dataGrid);
            dataGrid.dataProvider = new DataProvider();
        }
        public function getDataGrid() : DataGrid {
            return dataGrid;
        }
    }
}
```

Листинг 3. – Основная программа

```
import data.Product;
import view.Table;
import fl.controls.Button;
import fl.data.DataProvider;
import fl.controls.DataGrid;
```

```
var table : Table = new Table();
addChild(table);
var addButton : Button = new Button();
addButton.x = 5;
addButton.y = 375;
addButton.width = 100;
addButton.height = 20;
addButton.label = "Новый";
addButton.addEventListener(MouseEvent.CLICK, addNewProduct);
addChild(addButton);
var delButton : Button = new Button();
delButton.x = 110;
delButton.y = 375;
delButton.width = 100;
delButton.height = 20;
delButton.label = "Удалить";
delButton.addEventListener(MouseEvent.CLICK, deleteProducts);
addChild(delButton);

function addNewProduct(e : MouseEvent) : void {
    var dataProvider : DataProvider =
                            table.getDataGrid().dataProvider;
    var identity : int;
    if(dataProvider.length > 0) {
        identity = dataProvider.getItemAt(0).identity;
        for(var i : int = 1; i < dataProvider.length; i++) {
            if(dataProvider.getItemAt(i).identity >
                                                    identity) {
                identity =
                            dataProvider.getItemAt(i).identity;
            }
        }
        identity++;
    } else {
        identity = 1;
    }
    var product : Product = new Product();
    product.identity = identity;
    dataProvider.addItem(product);
}

function deleteProducts(e : MouseEvent) : void {
    var dataGrid : DataGrid = table.getDataGrid();
    for each(var item in dataGrid.selectedItems) {
```

```
            dataGrid.removeItem(item);
        }
    }
```

Задание 2

Написать программу для управления списком некоторых объектов (см. вариант). Предусмотреть обработку ошибочно введенных данных. Реализовать обработку данных, предусмотренных вариантом.

Вариант 1.

Список книг различных авторов (шифр книги, автор, название, год издания, количество страниц). В отдельное текстовое поле выводить автора самой свежей книги (или авторов через запятую).

Вариант 2.

Список заданий для различных проектов (шифр задания, название проекта, описание задания, планируемое время выполнения в часах, фактическое время выполнения в часах). Отдельно выводить название задания с наибольшим отставанием (или несколько таких заданий через запятую).

Вариант 3.

Список студентов факультета (номер зачетки, группа, фамилия, имя, отчество, пол, курс, средний балл). В отдельное многострочное текстовое поле выводить средний балл студентов по каждому курсу.

Вариант 4.

Каталог музыкальных произведений (идентификатор в каталоге, автор (группа), название, время звучания (в формате «mm:ss»), дата добавления в каталог. Отдельно выводить общее время звучания всех произведений, добавленных в каталог за последний месяц (вывод в формате «dd дней, hh:mm:ss»).

Вариант 5.

График отпусков сотрудников организации (код сотрудника, отдел, фамилия, имя, отчество, дата начала отпуска, дата окончания отпуска, заработная плата, размер отпускных). Последнее поле должно рассчитываться автоматически исходя из того, что за каждый день отпуска любой сотрудник получает определенный процент (единый для всех сотрудников) от своей зарплаты. Также подчитать в отдельном текстовом поле общую сумму отпускных по организации. В отдельное многострочное поле по каждому отделу привести промежуток времени, в который в отделе будет работать наименьшее количество человек.

Лабораторная работа №5
Загрузка внешних данных

Задание 1

Добавить в приложение предыдущей лабораторной работы возможность отображения изображения для каждого объекта из списка. Для этого добавить новое свойство в классе Product

Листинг 1. – дополнение класса Product

```
private var imageProp : String;
public function get image() : String {
    return imageProp;
}
public function set image(imageProp : String) : void {
    this.imageProp = imageProp;
}
```

Изменить размер сцены с 550x400 до 750x400. Изменить ширину dataGrid в классе Table на 740. Дополнить код созданием еще одной колонки:

Листинг 2. – дополнение класса Table

```
var imageColumn : DataGridColumn = new DataGridColumn();
imageColumn.dataField = "image";
imageColumn.headerText = "URL изображения";
dentityColumn.width = 200;
```

Добавить созданную переменную imageColumn в массив dataGrid.columns.

Создайте в библиотеке новый символ, внутри которого изобразите прямоугольник размера 100x100 с координатами x = 0, y = 0. Без внешней рамки темно-серого цвета с прозрачностью, равной 50%. Экспортируйте символ для ActionScript под именем BG. В начале кода в первом фрейме добавьте создание и инициализацию дополнительных объектов.

Листинг 3. – Изменение основной программы

```
var bg : BG = new BG();
bg.x = 0;
bg.y = 0;
bg.width = 750;
bg.height = 400;
```

```
var uiLoader : UILoader = new UILoader();
uiLoader.x = 5;
uiLoader.y = 5;
uiLoader.width = 740;
uiLoader.height = 390;
uiLoader.addEventListener(Event.COMPLETE, showImage);
var closeImageButton : Button = new Button();
closeImageButton.x = 725;
closeImageButton.y = 5;
closeImageButton.width = 20;
closeImageButton.height = 20;
closeImageButton.label = "x";
closeImageButton.addEventListener(MouseEvent.CLICK,
                                                  closeImage);
```

После создания таблицы и назначения слушателя события DataGridEvent.ITEM_EDIT_END добавьте слушателя еще одного события:

```
table.getDataGrid().addEventListener(
                          MouseEvent.DOUBLE_CLICK, openImage);
```

Добавьте функции-обработчики событий:

Листинг 4. – обработчики событий

```
function openImage(e : MouseEvent) : void {
    var url : String =
                        table.getDataGrid().selectedItem.image;
    if(url != null && url.length > 0) {
      uiLoader.load(new URLRequest(url));
      addChild(bg);
    }
}
function closeImage(e : MouseEvent) : void {
    removeChild(closeImageButton);
    removeChild(uiLoader);
    removeChild(bg);
}
function showImage(e : Event) : void {
    addChild(uiLoader);
    addChild(closeImageButton);
}
```

Задание 2

Добавьте возможность просмотра изображения во второе задание предыдущей лабораторной работы.

Самостоятельно добавьте возможность индикации текущего состояния загрузки изображения (с помощью компонента fl.controls.ProgressBar и события ProgressEvent.PROGRESS компонента fl.containers.UILoader). Не забудьте установить свойство source компонента ProgressBar равному соответствующему компоненту UILoader. В обработчике события ProgressEvent.PROGRESS следует воспользоваться методом setProgress() класса ProgressBar. Метод принимает два целочисленных параметра: количество загруженных байт; общее количество байт. Данную информацию можно узнать из класса, соответствующего событию, а именно через его свойства bytesLoaded и bytesTotal. При завершении загрузки необходимо вызвать метод reset() класса ProgressBar, чтобы индикатор загрузки был готов к следующей загрузке. Также необходимо обработать событие IOErrorEvent.IO_ERROR компонента fl.containers.UILoader, в обработчике которого как минимум убрать со сцены объекты классов BG и fl.controls.ProgressBar.

Лабораторная работа №6
XML

Задание 1

Добавить в приложение предыдущей лабораторной работы возможность импорта данных из xml файла:

Листинг 1. – файл products.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<products>
    <category>
      <name>Ноутбуки</name>
      <product image="/images/notebooks/acer.jpg">
           <name>Acer</name>
           <amount>15</amount>
           <price>995</price>
      </product>
      <product>
           <name>Toshiba</name>
           <amount>10</amount>
           <price>1295</price>
      </product>
      <product image="/images/notebooks/asus.jpg">
           <name>Asus</name>
           <amount>5</amount>
           <price>1995</price>
      </product>
    </category>
    <category>
      <name>Мобильные телефоны</name>
      <product image="/images/phones/ericsson.jpg">
           <name>Sony Ericsson</name>
           <amount>50</amount>
           <price>159</price>
      </product>
      <product>
           <name>Nokia</name>
           <amount>30</amount>
           <price>219</price>
      </product>
    </category>
    <category>
      <name>Фотоаппараты</name>
```

```
        <product image="/images/photo/nikon.jpg">
            <name>Nikon</name>
            <amount>3</amount>
            <price>2900</price>
        </product>
        <product image="/images/photo/canon.jpg">
            <name>Canon</name>
            <amount>3</amount>
            <price>3900</price>
        </product>
    </category>
</products>
```

Для этого необходимо внести в код следующие изменения:

Произвести изменения в функции addNewProduct, вынеся ее код в отдельную функцию appendProduct

Листинг 2. – функция appendProduct

```
function appendProduct(product : Product) {
    var dataProvider : DataProvider =
                                table.getDataGrid().dataProvider;
    var identity : int;
    if(dataProvider.length > 0) {
      identity = dataProvider.getItemAt(0).identity;
      for(var i : int = 1; i < dataProvider.length; i++) {
          if(dataProvider.getItemAt(i).identity >
                                                  identity) {
              identity =
                          dataProvider.getItemAt(i).identity;
          }
      }
      identity++;
    } else {
      identity = 1;
    }
    product.identity = identity;
    dataProvider.addItem(product);
}
```

В самой функции addNewProduct просто вызвать функцию appendProduct

Листинг 3. – функция addNewProduct

```
function addNewProduct(e : MouseEvent) : void {
    appendProduct(new Product());
}
```

Импорт из xml документа реализуем в отдельном классе Importer.

Листинг 4. – класс Importer

```
package model {
    import data.Product;

    public class Importer {
      private var productsProp : Array;

      public function Importer(file : String) {
           productsProp = new Array();
           var products : XML = new XML(file);
           for each(var category in products.category) {
                for each(var product in category.product) {
                     var p : Product = new Product();
                     p.category = category.name;
                     p.image = product.@image;
                     p.name = product.name;
                     p.amount = product.amount;
                     p.price = product.price;
                     productsProp.push(p);
                }
           }
      }

      public function get products() : Array {
           return productsProp;
      }
    }
}
```

Создать в основном коде приложения объект класса URLLoader, назначив ему соответствующий обработчик события.

Листинг 5. – добавление объекта класса flash.net.URLLoader

```
var urlLoader : URLLoader = new URLLoader();
urlLoader.addEventListener(Event.COMPLETE, importProducts);
```

```
function importProducts(e : Event) : void {
    var importer : Importer = new Importer(urlLoader.data);
    for each(var product in importer.products) {
      appendProduct(product);
    }
}
```

Добавить кнопку с соответствующим обработчиком

Листинг 6. – добавление кнопки «Импорт»

```
var importButton : Button = new Button();
importButton.x = 215;
importButton.y = 375;
importButton.width = 100;
importButton.height = 20;
importButton.label = "Импорт";
importButton.addEventListener(MouseEvent.CLICK,
                                        startImportProducts);
addChild(importButton);

function startImportProducts(e : MouseEvent) : void {
    urlLoader.load(new URLRequest("products.xml"));
}
```

Задание 2

Добавьте возможность импорта данных из XML файла в индивидуальный проект (задание 2 предыдущей лабораторной работы).

Самостоятельно добавьте возможность индикации текущего состояния импорта данных. Для этого можно использовать компоненты, использованные ранее.

Лабораторная работа №7
AIR

Задание 1

Импортируйте приложение предыдущей лабораторной работы для платформы AIR:

1. Откройте окно настроек публикации (File > Publish Settings)
2. На вкладке Flash в выпадающем списке Player выберите пункт Adobe AIR 1.1 и нажмите кнопку Settings рядом с ним.
3. В открывшемся окне AIR – Application & Installer Settings найдите строку Digital Signature и нажмите кнопку Change.
4. В окне Digital Signature создайте новую цифровую подпись (кнопка Create). Поля «Publisher name», «Organization unit» и «Organization name» заполните произвольными данными на Ваше усмотрение, в поля «Password» и «Confirm password» введите некоторый одинаковый пароль (который Вы не забудете). В разделе «Save as» выберите место для сохранения файла цифровой подписи. Нажмите кнопку OK.
5. После создания цифровой подписи в окне Digital Signature еще раз укажите свой пароль. Снимите галочку с пункта Timestamp.
6. В разделе Included files окна AIR – Application & Installer Settings добавьте файлы и папки, с которыми работает ваше приложение (импортируемый xml-файл, папку с изображениями). Нажмите OK.
7. Убедитесь в работоспособности ролика (Control > Test Movie)

Добавим в приложение возможность работать с базой данных. Создайте базу данных SQLite в файле `products.sqlite` (например, с помощью SQLite Навигатора), создайте в ней таблицу `products`, заполните ее некоторыми данными.

Листинг 1. – пример создания таблицы `products`

```
CREATE TABLE products (
    identity INTEGER PRIMARY KEY AUTOINCREMENT,
    category CHAR(100),
    name CHAR(200),
    amount INTEGER,
    price FLOAT,
    image CHAR(512)
);
```

Добавьте в основной код приложения необходимый импорт:

Листинг 2. – импорт классов для работы с базой данных

```
import flash.data.SQLConnection;
import flash.data.SQLStatement;
import flash.data.SQLResult;
```

Реализуйте чтение информации из базы данных:

Листинг 3. – чтение из базы данных

```
var folder : File = File.applicationDirectory;
var db : File = folder.resolvePath("products.sqlite");
var connection : SQLConnection = new SQLConnection();
connection.open(db);
var query : SQLStatement = new SQLStatement();
query.sqlConnection = connection;
query.text = "SELECT identity, category, name, amount, price,
                                        image FROM products;";
query.execute();
var result : SQLResult = query.getResult();

for each(var row : Object in result.data) {
    var product : Product = new Product();
    for(var column : String in row) {
      product[column] = row[column];
    }
    table.getDataGrid().dataProvider.addItem(product);
}

connection.close();
```

Заменим теперь функцию `appendProduct` на новую реализацию:

Листинг 4. – новая функция `appendProduct`

```
function appendProduct(product : Product) {
    var dataProvider : DataProvider =
                              table.getDataGrid().dataProvider;
    createProduct(product);
    dataProvider.addItem(product);
}
```

Добавим функцию `createProduct`:

Листинг 5. – функция `createProduct`

```
function createProduct(product : Product) {
    connection.open(db);
    query.clearParameters();
    query.text = "INSERT INTO products (category, name,
                                        amount, price, image)
                  VALUES (:category, :name, :amount, :price,
                                                    :image);";
    query.parameters[":category"] = product.category;
    query.parameters[":name"] = product.name;
    query.parameters[":amount"] = product.amount;
    query.parameters[":price"] = product.price;
    query.parameters[":image"] = product.image;
    query.execute();
    product.identity = connection.lastInsertRowID;
    connection.close();
}
```

Заменим также функцию `deleteProducts`:

Листинг 6. – новая функция `deleteProducts`

```
function deleteProducts(e : MouseEvent) : void {
    var dataGrid : DataGrid = table.getDataGrid();
    connection.open(db);
    query.clearParameters();
    query.text = "DELETE FROM products WHERE identity=?;";
    for each(var item in dataGrid.selectedItems) {
      query.parameters[0] = item.identity;
      query.execute();
      dataGrid.removeItem(item);
    }
    connection.close();
}
```

Для отслеживания изменения данных в таблице добавим еще один обработчик события `ItemEditEnd`:

Листинг 7. – обработка изменения записи

```
table.getDataGrid().addEventListener(
                DataGridEvent.ITEM_EDIT_END, updateProduct);

function updateProduct(e : DataGridEvent) : void {
    var product : Product = e.itemRenderer.data;
    connection.open(db);
    query.clearParameters();
    query.text = "UPDATE products SET category=:category,
                  name=:name, amount=:amount, price=:price,
                   image=:image WHERE identity=:identity;";
    query.parameters[":identity"] = product.identity;
    query.parameters[":category"] = product.category;
    query.parameters[":name"] = product.name;
    query.parameters[":amount"] = product.amount;
    query.parameters[":price"] = product.price;
    query.parameters[":image"] = product.image;
    query.execute();
    connection.close();
}
```

Задание 2

Добавьте возможность работы с базой данных в индивидуальный проект.