МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний авіаційний університет

Електронне видання

ТЕХНОЛОГІЇ ЕЛЕКТРОННИХ МУЛЬТИМЕДІЙНИХ ВИДАНЬ

Лабораторний практикум
для здобувачів вищої освіти ОС «Бакалавр»
Галузь знань: 18 Виробництво та технології
Спеціальність: 186 Видавництво та поліграфія
Освітньо-професійна програма:
Технології електронних мультимедійних видань

Київ 2022

УДК 004. 032.6(076.5)
ББК З 973.0р
Т 384

Укладачі: М.А. Мелешко, І.А. Гніденко, В.А. Ракицький

Рецензент: О.А.Бобарчук

Затверджено методично-редакційною радою факультету міжнародних відносин (протокол № _1_ / від _12_._01_.2023 р.).

Т 384 **Технології електронних мультимедійних видань:**
Лабораторний практикум / укладачі: М.А. Мелешко,
І.А.Гніденко, В.А. Ракицький – К., НАУ, 2022. – 77 с.

У лабораторному практикумі наведено опис основних операцій підготовки й створення електронних мультимедійних видань, а також завдання до виконання лабораторних робіт.

Для студентів Освітньо-професійна програма:
Технології електронних мультимедійних видань
Галузь знань: 18 Виробництво та технології
Спеціальність: 186 Видавництво та поліграфія

ЗМІСТ

ВСТУП

Лабораторні роботи виконуються згідно з програмою дисципліни «Технології електронних мультимедійних видань» для здобувачів вищої освіти ОС «Бакалавр» з метою напрацювання у практичних навичок створення електронних мультимедійних видань за допомогою видавничих систем і програмування, а також для набуття вмінь корпоративної роботи з документами.

Лабораторний практикум розроблено відповідно до вимог кредитно-модульної системи оцінки знань та передбачає проведення по чотири лабораторні роботи у 1 та 2 модулі та шість лабораторних робіт у 3 модулі.

В основу електронного формату даного практикуму покладені матеріали друкованого видання

В той же час, внесені зміни та доповнення відповідно до робочого навчального плану 2021 року та нової робочої програми навчальної дисципліни «Технології електронних мультимедійних видань». А також внесені корективи щодо використання програмного забезпечення у зв'язку з тим, що Flash, який широко використовувався в навчальному процесі на даний час не підтримується компанією-розробником.

Під час підготовки до виконання робіт, студенти мають ознайомитися з відповідним теоретичним матеріалом конспекту лекцій і рекомендованої літератури.

Результат роботи оформлюється у письмовому вигляді як реферат у файлі MS OFFICE з назвою: № групи-призвіще студента-№ лабораторного заняття (наприклад – "415-Петренко-02").

У висновках потрібно охарактеризувати компоненти програмного середовища та особливості створеного програмного продукту.

Звіти до лабораторних робіт повинні включати: назву роботи, мету її виконання, задачі, які необхідно при цьому вирішувати, і послідовність виконання роботи, із зазначенням інструментів, які були використані під час виконання кожного завдання. Окремі етапи роботи необхідно проілюструвати рисунками чи скріншотами. Обов'язковим елементом звіту є відповіді на контрольні запитання. В кінці звіту надаються висновки за результатами виконання лабораторної роботи.

МОДУЛЬ 1 «КОНЦЕПТУАЛЬНІ ОСНОВИ ПРОЄКТУВАННЯ ЕЛЕКТРОННИХ МУЛЬТИМЕДІЙНИХ ВИДАНЬ»

Лабораторна робота № 1.1

Дослідження технології створення градієнту

Мета:

- опанувати прийоми створення зображень;
- вивчити засоби створення градієнту.

Технічне і програмне забезпечення:

1. Персональний комп'ютер.
2. Операційна система Windows 7 і новіша.
3. Програмне середовище (по вибору здобувача): NanoFL, Lightspark, Ruffle, BlueMaxima's Flashpoint, HTML5.

Завдання:

1. Створення 2D-сцени, відповідно ло варіанту.
2. Розфарбування сцени різними градієнтними пензлями.

Основні теоретичні відомості:

Градієнт – це головний перехід від одного кольору до іншого. В наш час застосування градієнтів стало дуже популярним, оскільки їх використання може надати ефект об'ємності у прикладних програмах. В основі градієнта лежить інтерполяція між двома й більше кольоровими значеннями. Існує три основних види градієнтів: лінійний, конічний, радіальний.

Рис. 1. Відображення лінійного градієнту.

Лінійні градієнти (рис. 1) задаються двома колірними точками контролю й декількома точками зупинки (color stops) на лінії, що сполучає кольори цих точок.

Таким чином задані три кольори на трьох різних позиціях між двома точками контролю. Позиції точок задаються дійсними значеннями від 0 до 1, де нуль представляє собою першу контрольну точку, а 1 – іншу. Кольори між цими точками будуть інтерполювані.

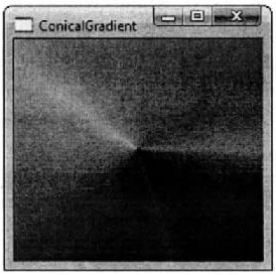


Рис. 2. Відображення конічного градієнту.

Конічний градієнт (рис. 2) задається центральною точкою та кутом. Поширення кольорів навколо центральної точки відповідає повороту часової стрілки.

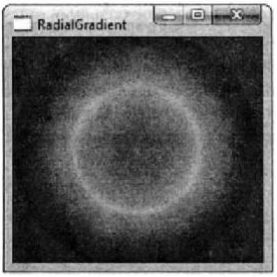


Рис. 3. Відображення радіального градієнту.

Радіальний градієнт (рис. 3) задається центральною точкою, радіусом та точкою фокуса. Центральна точка й радіус задають

коло. На поширення кольорів за межами точки фокуса впливає центральна точка або точка, що знаходиться всередині кола.

Порядок виконання роботи:

1. Оберіть тему у відповідності до варіанту.
2. Вивчіть документацію та обробіть матеріал.
3. Виконайте роботу.
4. Оформіть звіт.
5. Завершивши роботу, здайте електронний варіант звіту через мережу викладачеві і закрийте всі вікна.
6. Передайте роздрукований примірник звіту викладачеві та захистіть роботу не пізніше закінчення наступного лабораторного заняття.

Варіанти індивідуальних завдань (задає викладач).

Контрольні запитання:

1. Що таке градієнт?
2. Які існують види градієнтів?
3. Як задати лінійний градієнт для зображень?
4. Як задати конічний градієнт для зображень?
5. Як задати радіальний градієнт для зображень?

Лабораторна робота № 1.2

Дослідження типів даних мультимедійної інформації та засобів її обробки

Мета:

- ознайомитися з типами даних та інтерфейсом;
- вивчити засоби обробки мультимедіа інформації.

Технічне і програмне забезпечення:

1. Персональний комп'ютер.
2. Операційна система Windows 7 і новіша.
3. Програмне середовище NanoFL, Lightspark, Ruffle, BlueMaxima's Flashpoint, HTML5.

Завдання:

1. Створення «Морфінгу» (трансформація).
2. Побудова псевдо 3D куба.
3. Створення інтерактивної «Button».

Основні теоретичні відомості:

Рекомендації до виконання завдання 1:

1.1. Відкрийте новий файл, встановіть розмір файлу 150x150 пікселів і біле тло.
1.2. Виберіть інструмент Oval (овал) або натисніть «O» на клавіатурі.
1.3. Далі виберіть свої улюблені кольори для заповнення й креслення, як показано нижче.
1.4. Створіть коло (переконаєтеся, що ви перебуваєте на першому кадрі вашої флеш-анімації) за допомогою інструмента Oval, тільки що обраного на полотні, утримуйте клавішу «Shift» натиснутої й рухайтеся по полотну, щоб одержати правильне коло.

1.5. Після цього відцентруйте коло, рухаючи його стрілками на клавіатурі, у вас повинен вийти об'єкт начебто того, що зображено нижче, з точністю до кольорів, які ви використовували.
1.6. Створіть новий ключовий кадр, наприклад, на 20-му кадрі (див. нижче).
1.7. Далі, акуратно додержуючись всіх кроків, виберіть 20-й кадр на тимчасовій шкалі й видаліть його. Коло також буде вилучене тільки з 20-го кадру.
Виберіть інструмент Rectangle (прямокутник) або натисніть на клавіатурі «R».
Змініть колір, якщо хочете, щоб вони змінювалися в процесі морфінгу.
Утримуючи «Shift», рухайтеся по діагоналі щоб створити правильний квадрат, відцентруйте його стрілками.
1.8. Виділіть у зворотному порядку від кадру 21 до 1, як показано нижче.
1.9. Перевірте, що панель Frame (кадр) не відкрита. Зайдіть у меню Window (вікно) і виберіть Window (вікно) > Panels (панелі) > Frame (кадр) або натисніть CTRL+F, переконаєтеся, що панель Frame усе ще не відкрита або закрийте її за допомогою Ctrl+F.
1.10. Виберіть «Shape» (форма) зі спадаючим меню Tweening (проміжний), див. нижче. Інші опції залишіть за замовчуванням.
1.11. Зніміть виділення з кадрів, клацнувши на порожню область тимчасової шкали, здійснюється перехід від кадру 1 до кадру 20. Зелений колір вказує на те, що це не стандартний перехід, а перетворення форми (див. нижче).

Рекомендації до виконання завдання 2:

2.1. Створіть зображння кубика (рис. 4).
2.2. Далі виділяємо всі лінії й у верхньому меню вибираємо Insert>Convert to Symbol називаємо cub і вибираємо «Graphic».
2.3. У 30 фрейм вставте кейфрейм. Натисніть «Shift» і виберіть всі фрейми. Потім правою кнопкою миші натисніть у 30-му фреймі й виберіть Panels>Frame.

2.4. У вікні, що з’явилось, виберіть Tweening: Motion, Rotate:CCW, 1:times.
2.5. Все розфарбуйте і переглянете, що у вас вийшло.

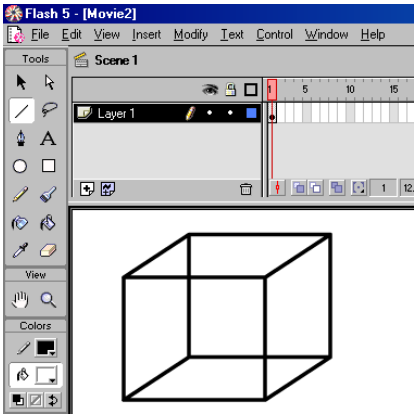


Рис. 4. Зображення псевдо 3D кубика.

2.6. Далі виділяємо всі лінії й у верхньому меню вибираємо Insert>Convert to Symbol називаємо cub і вибираємо «Graphic».
2.7. У 30 фрейм вставте кейфрейм. Натисніть «Shift» і виберіть всі фрейми. Потім правою кнопкою миші натисніть у 30-му фреймі й виберіть Panels>Frame.
2.8. У вікні, що з’явилось, виберіть Tweening: Motion, Rotate:CCW, 1:times.
2.9. Все розфарбуйте і переглянете, що у вас вийшло.

Рекомендації до виконання завдання 3:

3.1. Відкривайте меню Insert → New Symbol, або натисніть Ctrl+F8. У меню, що з'явився, даємо ім'я, і вибираємо тип об'єкта: Movie clip, Button (саме кнопка) і Graphic (просто картинка). Вибираємо Button, і називаємо but1. Розкривається вікно редагування цієї кнопки. Вище робочої області знаходиться основна частина програми флеш - ніби монтажерський стіл, кожний невеликий осередок - окремий кадр. У цьому випадку ми редагуємо кнопку, і в нас 4 прямокутники з підписами: Up,

Over, Down, Hit - вони позначають положення мишки стосовно нашої кнопки, відповідно звичайний вигляд кнопки, курсор над кнопкою, клік, і область кліку. Ліворуч від усіх цих прямокутників знаходиться шар Layer 1.

3.2. Виберіть шар, клацніть по прямокутнику з підписом Up і намалюйте звичайний вигляд кнопки. Можете намалювати коло, і помістити в ньому текст. Готово? Тепер натискаємо правою кнопкою по прямокутнику з написом Over і вибираємо в меню, що випадає, Insert Keyframe. Тепер поміняємо на кнопці, наприклад, колір. Insert Keyframe під Down і змінюємо колір і там.

3.3. Знайдіть, трохи вище ліворуч, напис Scene 1 | but 1 - це говорить про те, що ми редагуємо символ but1 у сцені Scene 1. Натисніть по Scene 1 і переходимо до основного фільму. Тепер, щоб вставити сюди свою кнопочку, «клікаємо» Ctrl+L або Window→Library, і одержуємо віконечко з назвою Library - це бібліотека, що містить всі символи, що входять у ваш фільм. Бачите but1? Перетягнете його куди-небудь на аркуш (у робочу область). Так ми помістили свою кнопку в основному фільмі. Збережіть файл куди-небудь на жорсткому диску File→Save. Тепер для перегляду того, що вийшло, тиснемо Control→Test Movie. От відкрилося вікно "предперегляду" - тобто так буде виглядати ваш фільм. Спробуйте підвести курсор до кнопки.

3.4. Далі потрібно в основному вікні редактора, клацнути правою кнопкою мишки на створеній нами кнопці, і в меню, що з'явиться, вибрати Actions. Серед доступних команд вибираєте Get Url і, у меню, що з'явився, пишете свій URL. Крім самої адреси, можна вказати ім'я вікна в рядку window (якщо використовуються фрейми). Там же можна вказати чи відкривати при клацанні по кнопці нове вікно (target_blank). Рядок Variables необхідний, якщо ви хочете передати зі свого ролику серверному скрипту які-небудь дані (можна вибрати метод POST або GET). Для нашої кнопки цей пункт не потрібний. Для публікації, скористайтеся File->Publish Settings. Де ви можете вибрати формат публікації (за замовчуванням це HTML) і зможете настроїти деякі параметри публікації. Після натискання Publish, Flash редактор створить вам однойменний HTML файл і *.swf файл (це flash-файл для публічного

перегляду, HTML ви можете сміливо правити, тільки обережніше зі спец. кодом).

Контрольні запитання:

1. Що таке морфінг?
2. Які є інструменти (засоби) у для здійснення морфінгу?
3. В яких випадках і навіщо потрібно створювати морфінг?
4. Яка мінімальна кількість кадрів потрібна для створення морфінгу?
5. Як створити псевдо 3D-фігуру у?
6. Які є інструменти (засоби) у для створення кнопки?
7. В яких випадках і навіщо потрібно створювати кнопки?

Лабораторна робота № 1.3

Дослідження технологій підготовки і публікації електронних мультимедійних видань в Internet

Мета:

- дослідження технологій підготовки і публікації електронних мультимедійних видань в Internet;
- придбання навичок щодо розробки банерної мережі та меню Web-сайту.
- дослідження технологій розробки анімації електронних мультимедійних видань в Internet;
- опанувати створення покрокової анімації Web-сайту.

Технічне і програмне забезпечення:

1. Персональний комп'ютер.
2. Операційна система Windows 7 і новіша.
3. Програмне середовище (вибирається з реомендованого в попепредніх роботах).

Завдання:

1. Створити банерну мережу з допомогою.
2. Розробити Internet-меню на.
3. Сформувати розміри сторінки на Web-сайті.
4. Створення найпростішої анімації для Web-сайту.

Основні теоретичні відомості:

Рекомендації до виконання завдання 1:

Найбільш популярний розмір банеру сьогодні становить **468x60** (у пікселях). Для простоти, але без шкоди для спільності, будемо робити банерну мережу із двох банерів. Для виконання даної лабораторної роботи вам знадобляться вже два створених банери (кнопки).

1.1. Підготуємо для них сцену проекту. Після **Ctrl+M**, установимо швидкість руху нашого фільму 1 fps (кадр за секунду), а розміри робочого поля (полотна) приведемо у відповідність зі стандартними розмірами банеру.

1.2. Перетягнемо мишкою наш перший банер на сцену в перший кадр із бібліотеки проекту й відцентруємо його. Він точно співпаде з робочою областю.

У термінології банер повинен, щонайменше, мати властивості кнопки. А кнопка підтримує гіпер-переходи, через **getURL()**. Після правого кліку на банері, виберемо Actions і для URL у цій лабораторній роботі поставимо просто діез #, щоб банер нас нікуди не повів. У кожному конкретному випадку Ви встановите відносну або абсолютну адресу гіперпереходу.

1.3. Виділимо цьому банеру 10 секунд життя в одному акті циклу. У нашому масштабі це 10 кадрів. 11-й кадр зробимо ключовим, внесемо на сцену другий банер, перший видалимо, і потім другий відцентруємо. За аналогією з попереднім кроком, дамо йому в якості адреси гіперпереходу діез # (або будь-яка інша адреса).

1.4. Дамо другому банеру покрасуватися 15 секунд в одному акті циклу, тому 25-й кадр зробимо ключовим. Протестуємо із секундоміром нашу двобанерну мережу - **Ctrl+Enter**.

1.5. За допомогою **Ctrl+Shift+F12** поміщаємо нашу двобанерну мережу в **HTML** контейнер. Після натискання на кнопку Publish у тій же директорії, де ми тримаємо проект fla, з'явиться однойменний html-файл.

За допомогою будь-якого HTML-Редактора витягнемо з файлу html нашу банерну мережу у вигляді об'єкта. Одержимо наступний код:

<OBJECT ...>

...

</OBJECT>

Даний код і є нашою двобанерною мережею. Його можна вставити в будь-яке місце робочого html-документа.

Рекомендації до виконання завдання 2:

2.1. Перейдемо до виконання другого завдання лабораторної роботи. Задайте своєму кліпу малу висоту й велику довжину. У даній роботі введено **550x25**. Колір тла зробіть чорним.

2.2. Другий етап - створіть всі текстові кнопки. Візьміть інструмент Text, натиснувши для цього T і надрукуйте що-небудь типу "На головну", "Посилання", "Гостьова", тощо. Далі вам потрібно перевести всі ці написи на кнопки - для цього треба виділити кожний текстовий елемент покажчиком (стрілочкою) і нажати **F8** або перейти до **Insert (вставка) >Convert to Symbol (перевести в символи)**. Проробіть це для кожного елемента, тільки переконаєтеся, що обрано пункт **"button"** "кнопка".

2.3. Тепер створіть щось на кшталт прямокутника. Тільки обов'язково робіть це на новому шарі. Переведіть цей символ у кліп, як ви зробили із кнопками в другому етапі, тільки цього разу обраний повинен бути пункт **"movie clip"** (кліп). Назвіть цей кліп trailer.

2.4. Тепер перетягнете значок **"trailer"** на основний робочий простір. Потім, виділивши кліп, перейдіть у пункт меню **Window>Properties**. Задайте значку ім'я **"trailer"**.

2.5. Закінчивши пункт 3 і 4, створіть новий шар і створіть у другому фреймі порожній ключовий кадр. Потім натисніть F5 у всіх інших шарах. Кликніть на другому фреймі й натисніть F7. Кликніть на першому фреймі й перейдіть у пункт меню **Window>Actions**. Далі - у верхньому правому куті повинна бути маленька чорна стрілочка (у версії Flash) або пара білих смуг (версія flash MX); кликніть на ній і перейдіть в **"Expert Mode"** (експертний режим). Тепер вставте наступний код:

```
mouse_x = _xmouse;
setProperty(_root.trailer, _x, mouse_x+
((getProperty(_root.trailer, _x)-mouse_x)/1.15));
```

Цифру можна міняти, як вам хочеться. Чим вона більше, тим швидше рух. Тому, якщо рух виробляється занадто повільно - просто поміняйте цифру.

Потім перейдіть у другий фрейм шару дій і вставте наступний код:

```
gotoAndPlay (1);
```

Натисніть **CTRL+ENTER** - протестуйте свій кліп, повинно працювати. От і все!

Рекомендації до виконання завдання 3:

Якщо потрібно зробити web-сторінку, або навіть цілий сайт, то в першу чергу, виникає питання про базові розміри

робочого поля проєкту. У даній роботі ширина й висота задаються у відсотках. При цьому, задовільні результати виходять для всіх основних розподільних здатностей екрану **(640X480, 800X600 і 1024X768)**.

1) Зробимо контейнер для Web-сайту. Після **Ctrl+M** установимо ширину **(width)** і висоту **(height) 750 px і 450 px**, відповідно. Натиснемо кнопку OK і на робочому полі проекту з такими розмірами помістимо сюжет сторінки.
2) Тепер після **File (файл) > Publish Settings**... (настроювання публікації...) **(Ctrl+Shift+F12)** переконаємося в наявності "пташки" у типах для HTML і перейдемо на однойменну вкладку.
3) На вкладці HTML встановимо розміри у відсотках (**Dimensions** (розмірність) > Percent (відсотки)), ширину (**Width**) і висоту (**Height**) по **100**. Масштаб **(Scale)** краще встановити в стан **Default (за замовчуванням) (Show all - "показати всі").** При таких установках, на різних розподільних здатностях екрану не буде лінійних перекручувань, але можуть з'явитися вгорі та внизу порожні області. Якщо для вашого сюжету не дуже критичне розтягання по вертикалі (наприклад, для текстів), то можна встановити масштаб у стан **Exact fit (точно відповідати).** Тоді навіть у повно-екранному режимі **(для IE F11)** ваш сюжет буде займати весь екран.
4) Нарешті, тиснемо кнопку **Publish** (публікувати) і одержуємо HTML-Контейнер у тій же директорії, де був розміщений fla-проект. Для IE у тілі тегів HTML-Контейнеру в будь-якому HTML-Редакторі потрібно додати наступні атрибути - **topmargin="0" leftmargin="0" rightmargin="0" bottommargin="0".** Якщо кліп має однорідне тло, то замість цього, у тілі тегів можна поставити атрибут **bgcolor**="колір тла кліпу". Такий спосіб повинен задовольнити будь-який браузер.
5) Якщо після виконання всіх цих процедур відкрити HTML-Документ, то весь кліп з сюжетом повинен займати повністю вікно браузера, причому при будь-якій розподільній здатності клієнтського екрана. Смуг прокручування бути не повинно. Якщо для вашого сюжету недостатньо площі, то можна у вихідному проекті збільшувати висоту **(height)** у пікселях і

одночасно пропорційно збільшувати висоту (Height) в HTML-Контейнері. Тоді **495 px** будуть відповідати **110%, 540 px - 120%**. При цьому з'явиться вертикальна смуга прокручування.

Рекомендації до виконання завдання 4:

Існує три види символів: анімація **(movie clip)**, кнопка **(button)** і зображення **(graphic):**

1.1. Зображення **(graphic)**, являє собою символ, що складається з єдиного кадру. Звідси його статична назва. Якщо символ дійсно являє собою статичний (без анімації) об'єкт, краще зробити його зображенням **(graphic**).

1.2. Кнопка **(button)**. В програмних засобах є спеціально пристосований під функції кнопки вигляд символу. У ньому є 4 кадри: **Up, Over, Down, Hit**, які містять наступні стани кнопок:

Up - звичайний стан кнопки.

Over - коли курсор мишки перебуває над кнопкою.

Down - коли курсор перебуває над кнопкою й натиснута клавіша миші.

Hit - звичайний стан для кнопки, що містить посилання, яке користувач вже відвідував.

1.3. Анімація **(movie clip)**. Це самий "повноцінний" тип символу. У ньому може бути будь-яка кількість кадрів. Символ цього типу може сприйматися як об'єкт типу **Movie в ActionScript**.

Символи можуть бути вкладеними поза залежністю від типу.

А також, існує два методи анімації - **покадрова** і **шляхом створення проміжних кадрів**.

Покадрова анімація

Це анімація, повністю складена із ключових кадрів, тобто Ви самі визначаєте, як вміст кадру, так і його "тривалість" (тобто скільки таких статичних кадрів буде займати зображення, рис. 5).

На часовій шкалі покадрова анімація виглядає в такий спосіб:

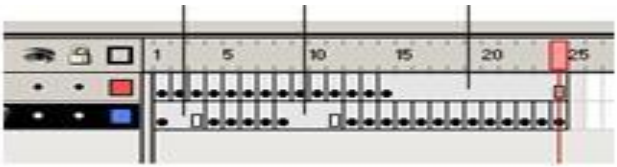


Рис. 5. Налаштування покадрової анімації.

Переваги:

Покадрова анімація дає Вам, у деякому сенсі, більший контроль над анімацією, і якщо ви досвідчений аніматор, Ви можете вигідно нею користуватися.

Це єдиний спосіб організувати зміну абсолютно незалежних зображень – слайд-шоу (наприклад, створюючи звичайний банер відповідними програмними засобами).

Недоліки:

Покадрову анімацію складно модифікувати. Особливо, якщо це не дискретний набір зображень, а зв'язана анімація. Доводиться модифікувати всі кадри.

Покадрова анімація займає досить великий обсяг, тому що доводиться зберігати інформацію про кожний кадр.

Контрольні запитання:

1. Що таке банер?
2. Які є інструменти (засоби) для створення та розміщення банеру на Web-сайті?
3. Які є правила розміщення банерів на Web-сайті?
4. Навіщо потрібне меню в програмі і чи на всіх Web-сайтах воно потрібне?
5. Які є інструменти (засоби) у для створення меню?
6. Які є інструменти (засоби) у для задання розмірів Web-сторінки?
7. Яка потрібна дозволяюча здатність екрану, для коректного відображення Web-сторінки?
8. Що таке анімація?
9. Які є інструменти (засоби) для здійснення анімації?
10. Скільки кадрів за секунду завантажується в сучасних фільмах?
11. Які існують види кадрів?
12. Які бувають шари?
13. Які колірні ефекти можна задати?
14. За допомогою яких інструментів можна задати колірні моделі?

Лабораторна робота 1.4

Дослідження технології підготовки та виробництва електронних книг в комп'ютерних видавничих системах

Мета та основні завдання роботи

Мета:

- дослідження технології підготовки та виробництва; електронних книг в комп'ютерних видавничих системах;
- вивчення прийомів роботи з текстом та рухом об'єктів в електронних мультимедійних ресурсах.

Технічне і програмне забезпечення:

1. Персональний комп'ютер.
2. Операційна система Windows 7 і новіша.
3. Програмне середовище (по вибору здобувача): NanoFL, Lightspark, Ruffle, BlueMaxima's Flashpoint, HTML5.

Завдання:

1. Імітація тіні. (на прикладі створення тіні для тексту).
2. Створення обертового тексту.
3. Створення руху об'єктів по траєкторії.

Основні теоретичні відомості:

Рекомендації до виконання завдання 1:

Як правило, тінь створюється в інших спеціальних редакторах, таких як Photoshop. Потім отриманий графічний об'єкт імпортується у відповідний файл. Однак обсяг swf-файлів при цьому виходять досить великим. З деякою втратою якості можна зробити тінь з використанням іншого програмного продукту, при цьому інформаційний обсяг буде в 6-10 разів менше.

1.1. Як приклад будемо робити тінь від тексту. Відкриємо і напишемо який-небудь текст великими жирними буквами довільного, але не чорного, кольору. Скопіюємо виділений текст Ctrl+C.

1.2. Додамо новий шар **Layer2**, перетягнемо його мишкою під перший **Layer1** Зробимо новий шар активним, вставимо туди раніше скопійований текст **Ctrl+V** і підберемо йому тіньовий

колір (рис. 6). Шар **Layer1** можна заблокувати, кликнувши мишкою під замочком.

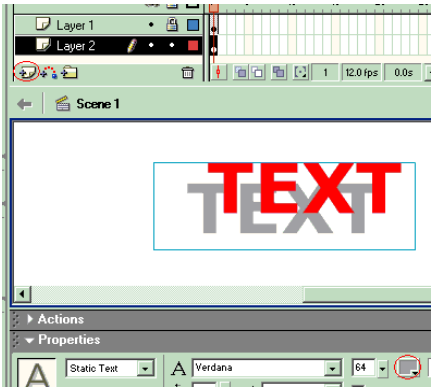


Рис. 6. Відображення ефекту тіні до об’єкту.

1.3. Тепер виділений текст другого шару **Layer2** переміщаємо клавіатурними стрілками так, щоб одержати імітацію тіні залежно від розташування джерела світла (в мене він праворуч угорі).

Рекомендації до завдання виконання 2:

2.1. Запустіть інструмент Text (текст), нажавши "T".

2.2. Тепер клацніть на сцену **(Stage)** і відкрийте меню **Properties** (властивості), для цього зайдіть в **Window** (вікно) **> Properties** (властивості), і встановите великий розмір тексту. Або такий, як вам буде зручніше. У даній лабораторній роботі встановлений **100**. Надрукуйте що-небудь. Перед наступним кроком відкрийте панель Info (інформація), нажавши **CTRL+I**. Запишіть куди-небудь ширину **(width)**, або просто добре запам'ятаєте її. У даному прикладі **460** пікселів. Запам'ятаєте це число - потім знадобиться.

2.3. Тепер натисніть F8 або зайдіть в **Insert (вставка) > Convert to Symbol (перетворити в символ)** і перетворіть ваш текст у графіку.

2.4. Зараз клацніть на кадрі 15 і натисніть F6, або зайдіть в **Insert (вставка) > Keyframe (ключовий кадр).**

2.5. Тепер клацніть на кадр 1, і поверніться до **Properties inspector** (інспектор властивостей) і під **"tween"** (перетворення) виберіть **"Motion"** (рух). Сірі до цього моменту кадри тепер повинні стати фіолетовими, і через них повинна проходити стрілка.

2.6. Поверніться до кадру 15 і клацніть на інструменті **Free Transform** (вільна трансформація), або натисніть **"Q"**. Клацніть на графіку й масштабуйте його до тонкої смужки, як на картинці нижче. Зробіть його шириною 1 піксель.

2.7. Як тільки зробили це, створіть новий ключовий кадр із кадру 16, так само, як і раніше. Тепер ідіть в Modify (змінити) > Transform (трансформація) > Flip Horizontal (відбити горизонтально).

2.8. Створіть ключовий кадр із кадру 31. Поки ви ще там, клацніть на зменшений графік і знову зайдіть в Properies inspector (інспектор властивостей). Там, де Width (ширина), впишіть число, що ви записали як ширину. У нашому прикладі знову 460, але тепер воно записано назад. Клацніть на кадр 16, зайдіть у меню properties (властивості) і виберіть "Motion" (рух) для Tween (перетворення).

2.9. Далі клацніть на кадрі **46** і створіть новий ключовий кадр. Візьміть інструмент **Free Transform** (вільна трансформація) знову й зменшите кадр до **1** пікселя (якщо не хочете використовувати інструмент трансформації, можете користуватися інспектором властивостей). Як тільки він зменшений, потрібно встановити перетворення руху від кадру **31** до **46**. Тому клацніть на кадр **31** і зайдіть в інспектор властивостей. Ще раз виберіть motion (рух) під Tween (перетворення).

2.10. Клацніть на кадр **47** і створіть новий ключовий кадр. Зайдіть в **Modify (змінити) > Transform (трансформація) > Flip Horizontal (відбити горизонтально).** Потім клацніть на кадр **62**, і створіть інший ключовий кадр. Збільште зображення назад до його звичайного розміру, що був **460 пікселів**. Але, щоб зробити його трохи більш цільним, нехай воно буде на декілька пікселів коротше. У даній лабораторній роботі вийшло **457**. Тепер все повинно виглядати приблизно так само, як і на малюнку нижче.

2.11. Останній крок - клацніть на останньому кадрі, і зайдіть в Window (вікно) > Actions (дії), щоб відкрити меню дій. Уведіть туди наступний код.

gotoAndPlay(1);

Тепер протестуйте кліп - зайдіть в **Control (керування) > Test Movie (тестувати кліп)** або натисніть **CTRL+ENTER**. Тест повинен продовжувати обертатися, й це виглядає дуже непогано.

Рекомендації до виконання завдання 3:

3.1. Для початку створіть новий файл, клацніть на **Add guide layer** (додати напрямний шар) на **Timeline** (тимчасовій шкалі), як показано нижче в червоному колі. Тим самим ви додасте напрямний шар (у червоному прямокутнику внизу), за замовчуванням у нього буде таке ж ім'я, як і на картинці нижче.

3.2. Тепер вам потрібно створити новий символ, що буде переміщуватися по шляху, що ми створимо пізніше. Він може бути чим завгодно - кулькою, бджолою, метеликом, літаком тощо (рис. 7). Намалюйте об'єкт і перетворіть його в символ, нажавши на клавіатурі **F8**, або зайшовши в **Insert (Вставка) > Convert to symbol (перетворити в символ).**

Рис. 7. Відображення літачка.

3.3. Виберіть напрямний шар, створений вами раніше, візьміть інструмент pen (перо)  , або натисніть 'P' на клавіатурі, щоб вибрати його, і намалюйте будь-яку траєкторію (рис. 8), тільки акуратно, тому що саме по ній ваш символ буде рухатися при анімації. Все стане зрозуміло, якщо подивитися на малюнок нижче. Клацніть, скажемо, на 100-**й кадр** і додайте ключові кадри на обох шарах, нажавши **F6** на клавіатурі.

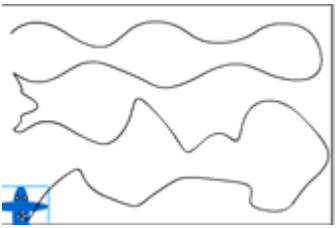


Рис. 8. Задавання траєкторії польоту літачка.

3.4. Зараз перетягнете створений вами символ на сцену шару 1 у кадр 1, якщо він ще не там. Ви побачите маленький хрестик, показаний нижче в червоному прямокутнику, вам потрібно клацнути на цей хрестик і перетягнути його в початок шляху. При перетаскуванні хрест перетвориться в маленький кружок, як показано стрілкою, і буде видний тільки контур символу. Вам потрібно сумістити кружок з початком траєкторії, вони "схопляться" один за одного, як показано нижче червоною стрілкою.

3.5. Тепер перейдіть до іншого (у цьому випадку до - 100-го) кадру й повторіть те ж саме, але цього разу з кінцем шляху.

3.6. Після того, як з'єднаєте шлях із символом, виберіть весь шар 1, клацнувши куди-небудь всередині нього, див. нижче:

3.7. Зайдіть в **Window (Вікно) > Modify (змінити) > Frame (кадр),** щоб відкрити палітру кадрів і вибрати те, що показано на картинці нижче.

Контрольні запитання:

1. Які спецефекти можна створювати у відповідному програмному забезпеченні?
2. Які є інструменти (засоби) для створення тіні до тексту?
3. Які є інструменти (засоби) у для обертання тексту?
4. Які є інструменти (засоби) для створення руху об’єкту?
5. Що таке кадр?
6. Яка мінімальна кількість кадрів необхідна для створення руху об’єкту по заданій траєкторії?
7. Яка мінімальна кількість шарів необхідна для створення руху об’єкту по заданій траєкторії?

МОДУЛЬ 2 «ТЕХНОЛОГІЇ ПІДГОТОВКИ ТА ВИРОБНИЦТВА ЕЛЕКТРОННИХ МУЛЬТИМЕДІЙНИХ ВИДАНЬ»

Лабораторна робота № 2.1

Дослідження технології обробки звукової інформації

Мета:

- засвоєння технології створенням звукових ефектів;
- імпортування звукових ефектів;
- навчитися працювати з основними властивостями звуку;
- придбання навичок щодо запису фрагментів електронного мультимедійного видання на компакт-диск.

Технічне і програмне забезпечення:

1. Персональний комп'ютер з тактовою частотою процесора 2Ггц і вище.
2. Операційна система Windows 7 і новіша.
3. Програмне середовище (по вибору здобувача): NanoFL, Lightspark, Ruffle, BlueMaxima's Flashpoint, HTML5.

Завдання:

1. Створити музичний супровід.
2. Створити mp3 файл із кліпом swf всередині.
3. Виконати імпорт звуків.

Основні теоретичні відомості:

Рекомендації до виконання завдання 1:

1.1. Необхідність у звуці виникає у двох випадках: по-перше, якщо потрібно створити звукове тло для якоїсь довгої сцени, наприклад, для сходу сонця, і по-друге, якщо ви хочете, щоб відео-ролик реагував різними звуками на натискання кнопок і наведення курсору мишки на його активні зони.

Спочатку розглянемо випадок, коли нам потрібно створити фоновий (потоковий) звук, що супроводжує яку-небудь подію довгої сцени. Для цього ми візьмемо готовий wav-файл і

імпортуємо його в відео відео-ролик. Ще раз акцентуємо увагу на тому, що поки не імпортовано жодного звуку в сцену, ніякої роботи зі звуком бути не може.

1.2. Після того, як звуковий файл успішно імпортований, створимо новий шар, що так і назвемо **“wav”,** і кликнемо курсором мишки на той його кадр, з якого ми хочемо почати звучання музики. У нашому випадку це буде перший кадр. Зробимо його опорним **(F6)**. Відкриємо пункт меню **Window/Panels/Sound** і ще раз натиснемо на цей кадр. У рядку **“Sound”** цього вікна перед нами з'явиться список всіх імпортованих звуків, з яких варто вибрати той, котрий нам потрібний. Трохи нижче виводяться всі параметри звукового файлу, а в рядку **“effects”** ви можете вибрати ефект відтворення даного звуку.

1.3. Розглянемо можливі ефекти:

Left Cannel – грає тільки ліва колонка,

Right Cannel – грає тільки права колонка,

Fade Left to Right - гучність повільно переходить із лівої колонки в праву,

Fade Right to Left - гучність повільно переходить із правої колонки в ліву,

Fade in – гучність наростає,

Fade out – гучність спадає.

В Flash існує також можливість редагування цих ефектів. Наприклад, можна зробити так, щоб гучність у лівій колонці протягом перших двох секунд звучання мелодії збільшилася на **25%**, а потім залишалася незмінною. Особливо цінний цей ефект, коли потрібно, щоб до кінця мелодія поступово затихала, що ми й зробимо. Для цього застосуємо ефект **Fade out**, і за допомогою кнопки edit відредагуємо звук таким чином, щоб звучання мелодії затихало в останні 3 секунди.

1.4. У поле Sync поставимо значення Start і наш Flash-ролик готовий. Тепер важливо відзначити деякі особливості, пов'язані з використанням звуку. Якщо ви захочете завантажити звуковий файл із мелодією на 3 хвилини, а ролик іде всього 5 секунд, то не турбуйтеся про це, зупинка відбудеться автоматично. Більше того, якщо в останньому кадрі звук не зупинити, то з кожним новим прокручуванням ролику породжується ще одну копію звуку, що буде накладатися на

попередню. Тому, час прокручування ролика повинен збігтися з часом звучання музики у звуковому файлі. Інакше, потрібно хоча б зупинити звук, вказавши в останньому кадрі в поле **Sync** значення **Stop**.

1.5. Тепер розглянемо, як можна створити звук. Для початку імпортуємо всі необхідні звукові файли, потім створимо всі необхідні кнопки. Після цього зайдемо в редактор кожної кнопки, і в кадрі відповідному оброблювачеві події **Down** за аналогією задамо необхідні параметри у вікні **Window/Panels/Sound:** Сам по собі програмний продукт є чудовим інструментом для створення досить складної мультиплікації в Інтернеті. Використання ж звукових елементів відкриває нові величезні можливості для творчості.

Рекомендації до виконання завдання 2:

2.1. Виберіть mp3, що ви хочете переробити. Тепер ми повинні конвертувати mp3 у формат wav. Для цього використовуємо, наприклад, WinAmp.

2.2. Запустіть WinAmp і зайдіть в Prefences > Plug-ins > Output і виберіть "Disk Writer Plug-in". Натисніть "Configure", виберіть директорію, у якій буде записаний wav-файл. (Пам’ятайте, що mp3 довжиною 3 хвилини займе приблизно 40Mb на вашому диску!) Тепер натисніть "Play" (ВИМКНІТЬ КНОПКУ "ПОВТОР" ("Repeat")) Ви нічого не почуєте під час програвання. Тепер у вас є wav-файл.

2.3. Створіть новий файл і збережіть його. Тепер нам треба імпортувати wav-файл в flash. Виберете в меню File > Import. Знайдіть ваш wav-файл і натисніть OK (якщо wav-файл великий, то вам доведеться трохи почекати). Тепер натисніть правою клавішею мишки на перший кадр і виберіть "Properties".

Виберіть "Sound". Тепер виберіть ваш звуковий файл

2.4. Тепер потрібно вибрати File→Publish settings. Виберіть "Formats". Скасуєте "use default names". У текстовому полі змініть розширення на "ім'я_вашого_файлу.mp3".

2.5. Далі виберіть закладку, натисніть кнопку "set" залежно від того, який тип синхронізації Ви вибрали раніше (у властивостях кадру 1). Виберіть такі установки, які хочете (стандартні установки для mp3: стерео, 128Kbps, Якість: best). Тепер

натисніть "Publish" (і почекайте... якщо у вас повільний комп'ютер, то доведеться чекати дуже довго).

2.6. Ну от майже й усе. Тепер тільки треба повідомити тим, хто буде слухати ваш mp3, що в ньому убудована крута анімація в flash. Це ми зробимо в "ID3 Info".

Запустіть WinAmp і натисніть праву клавішу мишки на заголовку й виберіть "file info" (або натисніть Alt+3). У поле "Artist" введіть "Rename 2 ^Title.swf^ u have", у поле "Album": "2 have the Flash plug-in", в "Comment": "http://www.Macromedia.com".

2.7. З допомогою програми Nero реалізуйте технологію запису цього фрагменту на компакт-диск.

Рекомендації до виконання завдання 3:

3.1. Натисніть **File>Import**. Виберіть файл на вашому вінчестері.

3.2. **Перегляд звуків**, це тут Window>Library.

3.3. **Видалення звуків**, зайдіть Window>Library. Правий клік Delete.

3.4. **Повтор звуків**, після того як ви вставите звук, натисніть на настроювання. Уведіть 999 де написане Loops:.

3.5. **Додавання звуків у кліп**, створіть фрейм. У його опціях натисніть "sounds" і виберіть потрібний звук.

3.6. **Вимкнення звуків**, натисніть Control>Mute Sounds.

3.7. **Додавання звуків у кнопки:**

а) Виберіть кнопку.

б) Правий клік, і у фреймі down вставте звук.

в) Зайдіть в опції звуків. Виберіть і тисніть ОК.

Контрольні запитання:

1. Який має вигляд та якою формулою задається найпростіший звуковий сигнал?
2. Які є алгоритми стиснення звукових сигналів?
3. Які є інструменти (засоби) для підключення звукового супроводу?
4. Який алгоритм стиснення звукового сигналу бажано використовувати при створенні Web-сторінки?
5. Як створити *.mp3-файл?
6. У чому полягає алгоритм стиснення звукового сигналу, котрий зберігається *.mp3-файлі?
7. Які є інструменти (засоби) для імпортування звуків?

Лабораторна робота № 2.2

Дослідження навігаційних структур при впровадженні гіпермедіа технологій

Мета: оптимізація роботи публікацій для WEB.

Технічне і програмне забезпечення:

1. Персональний комп'ютер з тактовою частотою процесора 2Ггц і вище.
2. Операційна система Windows 7 і новіша.
3. Програмне середовище (по вибору здобувача): NanoFL, Lightspark, Ruffle, BlueMaxima's Flashpoint, HTML5.

Завдання:

1. Створення "прелоадера".

Основні теоретичні відомості:

У даній лабораторній роботі описані тільки конструкції, що дозволяють реалізувати будь-який прелоадер (рис. 9), наприклад з лінійкою завантаження тощо, але необхідно враховувати, що всі графічні фрагменти завантажуються в першому кадрі, тобто чим він більше, тим довше буде вантажитися сам прелоадер!

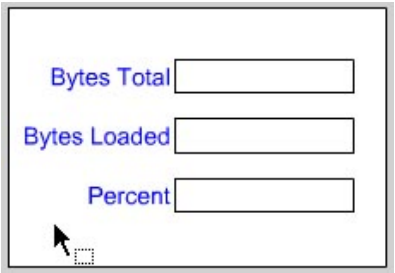


Рис. 9. Відображення базового прелоадеру.

А чи потрібний взагалі прелоадер? У режимі TestMovie у меню Debug виберіть або встановить свою швидкість завантаження

файлу (урахуйте, що реальна швидкість завантаження в користувача буде набагато нижче зазначеної):

У меню View вибираємо пункт Show Streaming:

Тестуємо ролик на різних швидкостях завантаження:

В інформацію, що виводить для клієнта прелоадер, можна включити поточну й середню швидкість завантаження (bps), зразковий час до закінчення завантаження ролику (remainig time) і лінійку візуалізації завантаження. Але варто враховувати, що всі ці художні «надмірності» збільшать і час завантаження самого прелоадера, тому їх бажано застосовувати для покращення очікування користувача, коли ролик і без того великий.

Рекомендації до виконання завдання:

1) Збільшимо розміри документа до 500 на 300 px (або який там вам подобається? Швидше за все по розміру основного фільму!):
2) Розміщаємо три написи: Current bps (поточна швидкість), Average bps (середня швидкість) і Remainig time (залишилося часу). Далі поміщаємо на Scene1 три порожні динамічні написи для виводу значень і призначаємо імена змінних для них, наприклад: CurrentField, AverageField і RemainigField:

   Для більшої "краси" і розваги користувача вставимо в наш прелоадер годинники для чого створимо в TimeLine папку (Folder, рис. 10), а в ній ще 2 шари:

Рис. 10. Відображення радіального градієнту.

3) Тепер створимо "лінійку" процесу завантаження (рис. 11). Для цього намалюємо витягнутий по горизонталі прямокутник, контур залишимо як є, а заливку перетворимо на символ. Потім трохи зменшимо символ і змістимо центр символу вліво (щоб

лінійка збільшувалася ліворуч/праворуч). Запишемо або запам'ятаємо довжину символу:
У розділі Instance Name назвемо символ Line. Тепер наш прелоадер повинен виглядати приблизно так:

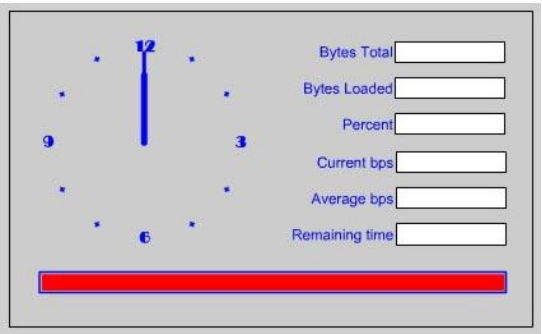


Рис. 11. Відображення радіального градієнту.

На цьому з "малюванням" закінчено, необхідно написати кодове супроводження.

4) У першому кадрі шару PLD розміщуємо наступний код:

Line._width = 0;

Встановлюємо довжину "лінійки", що відображає процес завантаження рівною 0;

Total = _root.GetBytesTotal();

Привласнюємо змінній Total розмір файлу в байтах;

TotalField = Total;

Виводимо значення цієї змінної на екран;

Cur = 0;

Обнуляємо значення змінної, в якій зберігається системний час у секундах;

Load = 0;

Обнуляємо значення змінної, в якій зберігається кількість завантажених байтів ролику;

E=0;

Обнуляємо значення змінний, в якій зберігається час процесу завантаження ролика в секундах.

5) У другому кадрі того ж шару пишемо:

B = new Date();

Змінній B привласнюємо значення системного часу;

Old = Cur;

Зберігаємо значення часу в секундах;

Cur = B.GetSeconds();

Одержуємо нове значення системного часу;

if (Cur <> Old) {

Якщо значення збереженого й нового системного часу не збігаються (пройшла 1 секунда) виконуємо наступний код:

E = E + 1;

Збільшуємо на 1 значення змінної, в якій зберігається час із початку завантаження ролику в секундах;

OldLoad = Load;

Зберігаємо значення кількості завантажених байтів;

Load = _root.GetBytesLoaded();

Одержуємо нове значення кількості завантажених байтів;

if (E <> 0) {Aver = int(Load / E);};

Якщо час завантаження ролику в секундах не дорівнює 0, то змінній Aver привласнюємо ціле значення відношення завантажених байтів, що вчасно надійшли з початку завантаження в секундах (середня швидкість завантаження ролику в байтах/сек.);

if (Aver <> 0) {Remain = int(Total / Aver);};

Якщо значення змінної середньої швидкості завантаження не дорівнює 0, то змінній Remain привласнюємо ціле значення відношення розміру ролику в байтах до середньої швидкості завантаження (зразковий час до закінчення завантаження в сек.);

CurrentField = Load - OldLoad;};

Виводимо в текстове поле CurrentField кількість завантажених байтів за останню секунду; Кінець оператора if;

RMin = int((Remain - E) / 60);

Змінної RMin привласнюємо ціле значення кількості хвилин до закінчення завантаження;

RSec = int(Remain - E - (RMin * 60));

Змінної RSec привласнюємо ціле значення кількості секунд до закінчення завантаження (за винятком цілих хвилин);

RMin = RMin + " m";

Додаємо позначення хвилин...

RSec = RSec + " s";

...і секунд;

if (length(RMin) < 4) {RMin = "0" + RMin;};

Якщо значення хвилин менше 10, додаємо попереду символ "0" (для краси);

if (length(RSec) < 4) {RSec = "0" + RSec;};

Те ж саме для секунд;

RemainingField = RMin + " : " + RSec;

Виводимо значення хвилин і секунд, розділених знаком ":";

LoadField = Load;

Виводимо кількість завантажених байт;

Percent = int(Load / Total * 100);

Змінній Percent привласнюємо відсоток завантаження ролику;

_root.Line._width = Percent * 4.42;

Змінюємо довжину символу Line відповідно до процентного значення завантаження ролику. 4.42 - коефіцієнт масштабування. Ми його запам'ятовували, коли малювали лінійку завантаження;

if (Percent < 100) {AverageField = Aver;};

Якщо ролик не завантажений повністю, то виводимо середнє значення швидкості завантаження;

PercentField = Percent + " %";

Виводимо значення процентного завантаження.

6) Третій кадр містить єдиний рядок:

if (load < Total){gotoAndPlay (2);};

Контрольні запитання:

1. Що таке "прелоадер"?
2. Які є інструменти (засоби) для створення "прелоадеру" на Web-сайті?
3. Які є правила розміщення "прелоадерів" на Web-сайті?
4. Навіщо потрібний "прелоадер" в програмі і чи на всіх Web-сайтах він потрібний?
5. Які є правила створення Web-сайтів?
6. Які є правила розміщення інформації на Web-сторінці?
7. Яка потрібна дозволяюча здатність екрану, для коректного відображення Web-сторінки?

Лабораторна робота № 2.3

Дослідження технологій підготовки інтерактивних сценаріїв

Мета: дослідження інтерактивності з використанням ActionScript

Технічне і програмне забезпечення:

1. Персональний комп'ютер з тактовою частотою процесора 2Ггц і вище.
2. Операційна система Windows 7 і новіша.
3. Програмне середовище (по вибору здобувача): NanoFL, Lightspark, Ruffle, BlueMaxima's Flashpoint, HTML5.

Завдання:

1. Створити інтерактивний фотоальбом вручну.
2. Створити інтерактивний фотоальбом, використовуючи автоматизоване написання скрипта.

Основні теоретичні відомості:

ActionScript являє собою мову написання сценарію - набору інструкцій, які управляють елементами фільму. Сценарії ActionScript можуть бути вбудовані у фільм або зберігатися в зовнішньому текстовому файлі з розширенням AS.

При вбудовуванні сценарію у фільм його можна впроваджувати в різні частини фільму. Точніше, сценарії ActionScript можуть містити ключові кадри, екземпляри кнопок і екземпляри MovieClip. Відповідно сценарії називаються сценаріями кадру (Frame Action), сценарієм кнопки (Button Action) і сценарієм кліпу (MovieClip Action).

Сценарії ActionScript виконуються при настанні певних подій, ініційованих користувачем або системою. Механізм, що вказує програмі Flash, який оператор варто виконати при настанні тої або іншої події, називається оброблювачем подій.

ActionScript має власний синтаксис, багато в чому схожий із синтаксисом JavaScript. Одним з основних понять ActionScript є Actions - команди, які видають інструкції під час виконання SWF-файлу. Наприклад, gotoAndStop() відсилає відтворюючу голівку (Playhead) на певний кадр або мітку. Від слова Actions і

відбувається назва мови - ActionScript (дослівно - сценарій дій). З більшістю понять цієї мови ми ознайомимося на конкретних прикладах.

Рекомендації до виконання завдання 1:

Продемонструємо використання кнопок для керування фотоальбомом - створимо набір фотографій і додамо дві кнопки, які будуть перегортати фото вперед та назад.

Помістимо на основній монтажній лінійці перше фото й додамо кнопку зі стандартного набору. Для доступу до потрібної папки варто виконати команду Windows => Control Panels Common Libraries => Buttons.

У результаті виконання даної команди з'явиться панель, що містить великий набір заздалегідь намальованих кнопок різних типів. Виберемо, наприклад, Key Buttons (кнопки, схожі на клавіатурні клавіші), відкриємо відповідну папку, виберемо кнопку key-left і створимо екземпляр даної кнопки (шляхом переміщення її на сцену).

Зверніть увагу, що, коли ви встановлюєте другу кнопку (key-right) на сцені й переміщаєте її, щоб установити на одному рівні з першою кнопкою, програма дає підказку (пунктирну лінію), що дозволяє точно позиціонувати кнопку.

Для того щоб додати сценарій, необхідно викликати редактор ActionScript по команді Window =>Development Panels => Actions або нажавши клавішу F9. Якщо ви збираєтеся часто писати скріпти, то цю клавіатурну команду варто запам'ятати. У результаті з'явиться редактор ActionScript.

Якщо ви проекспериментуєте, виділяючи різні елементи на сцені, одночасно відслідковуючи повідомлення в панелях редактора ActionScript, то виявите, що програма підказує, на який елемент можна «вішати» код. Якщо виділити на сцені кадр, то у верхньому лівому куті панелі редактора з'являється напис Actions-Frame, якщо клацнути по кнопці, то з'явиться напис Actions-Button, тобто програма підказує, що вводиться код, що, буде ставитися до сценарію кнопки. А якщо виділити фотографію, то в полі, призначеному для введення скрипта, з'явиться повідомлення: Current selection cannot have actions applied to it (до даного виділеного об'єкта не можна застосувати сценарій).

Ми будемо привласнювати сценарій кнопці. В останніх версіях ActionScript є можливість писати централізований код, тобто код, що розміщений в одному місці, і така можливість дозволяє краще розбиратися в більших програмах. Однак у простих прикладах (які ми й розглядаємо) присвоювання сценарію кнопці цілком припустимо.

Отже, для кнопки зі стрілкою вліво нам потрібно формалізувати наступний сценарій: «Якщо кнопка відпускається на деякому кадрі, то із цього кадру необхідно перейти на попередній кадр». Відповідно до синтаксису мови Action Script це буде виглядати в такий спосіб:

```
on (release)
{
        prevFrame();
}
```

У першому рядку записаний оброблювач подій кнопки on( ), що має формат:

```
on (event)
{
        текст сценарію
}
```

Тут event - це назва події, у розглянутому випадку release (відпускання кнопки); може також відслідковуватися така подія, як press (натискання кнопки), і інші події, які ми розглянемо пізніше.

Функція prevFrame - це функція безумовного переходу, що переміщає відтворюючу голівку в попередній кадр. Вона перебуває всередині оброблювача подій, тобто може виконуватися тільки у випадку настання описаної події.

Аналогічно на другу кнопку повісимо код, що забезпечує перехід до наступного кадру:

```
on (release)
{
        nextFrame();
}
```

Тепер додамо кілька ключових кадрів так, що в них перенесемо створені в першому кадрі кнопки, і помістимо в знову створені кадри потрібні фотографії.

Якщо запустити на виконання створений фільм, то кадри будуть безупинно програватися один за іншим, а отже, насамперед нам потрібно дати команду «Стоп» на першому кадрі. Для цього першому кадру додамо відповідну команду.

Зверніть увагу: команда привласнюється вже не кнопці, а кадру. Той факт, що кадру привласнений сценарій, відзначається на основній монтажній лінійці - у позначенні кадру над жирною крапкою з'являється мала літера «а».

У результаті ми одержали фільм.

Якщо в нас у фотоальбомі всього кілька кадрів, то двох кнопок - «Уперед» і «Назад» - цілком достатньо, але якщо набір фотографій великий, то бажано мати ще й кнопки, що відсилають у початок і кінець фільму. У наступному прикладі ми додамо відповідні кнопки: «У перший кадр» і «В останній кадр». Вибрати кнопки підходящої мнемоніки можна зі стандартної бібліотеки кнопок з папки Circle Buttons.

На запропонованому прикладі ми ознайомимося із ще однією командою gotoAndStop( ), що дозволяє перейти до потрібного кадру з наступною зупинкою.

У випадку з фотоальбомом з п'яти кадрів в останню кнопку додамо сценарій переходу «В останній кадр»:

```
on (release)
{
        gotoAndStop(5);
}
```

В результаті одержимо фільм.

Рекомендації до виконання завдання 2:

Дотепер ми набирали всі команди вручну, однак панелі редактора ActionScript надають цілий ряд сервісів для автоматизованого написання скрипта. Розглянемо ці можливості.

Панель редактора Action Script дозволяє вибирати, перетаскувати, перерозподіляти й видаляти команди.

Покажемо, як можна написати той же скрипт для кнопки «Уперед» в автоматизованому режимі. Вибравши папку Movie Clip Control можна одержати доступ до оброблювача подій on, а далі потрібно або двічі клацнути по відповідному пункті, або перетягнути вираження на поле скрипта в режимі Drag and Drop. В результаті на робочому полі з'явиться необхідний вираз і

підказка то ви вибираєте з меню потрібну команду й вираз завершується автоматично. Як видно з меню, можна вибрати не тільки умову, пов'язану з екранними кнопками, - можна, також, вибрати з меню пункт keyPress “<Left>”, або keyPress “<Right>”, що відповідає натисканню клавіатурних клавіш (стрілка вліво, стрілка вправо), тобто є можливість створити фотоальбом, що буде «перегортатися» за допомогою клавіш клавіатури.

Задіємо команду keyPress “<Right>”, потім перейдемо в папку Timeline Control, виберемо команду nextFrame і перемітимо її на робоче поле.

Для кнопки, що переводить фільм у початок фотоальбому, як подія можна вибрати з меню натискання клавіші Home, а далі (рис. 12) перетягнути на поле команду gotoAndStop, у результаті чого з'явиться ще одна підказка до даної команди.

Стрілочки в підказці дозволяють переглянути різні варіанти синтаксису. Програма надає два варіанти (рис. 12), тобто пропонує задати сцену й кадр або тільки кадр. У нашім випадку досить вказати тільки кадр (Frame). Якщо назва сцени пропущена, то за замовчуванням здійснюється перехід до кадру поточної сцени.

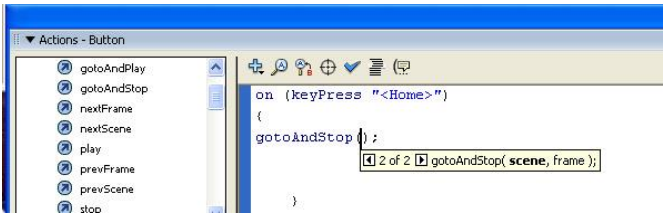


Рис. 12. Підказка до команди

Після того як ми призначимо всім кнопкам відповідні кнопки клавіатури, одержимо фільм, де перегортання фотографій буде відбуватися із клавіатури, а натискання мишки на екранні кнопки не буде викликати ніяких наслідків.

Чи можна забезпечити сценарій, при якому різні події будуть приводити до одних і тих же дій? Виявляється, можна - для цього в обробловачі подій on необхідно перелічити список найменувань подій. Якщо поставити кому в списку подій після першої події, то програма сама запропонує вам меню (рис. 13).

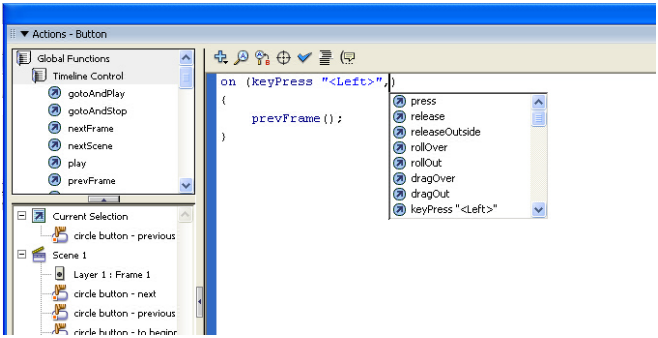


Рис. 13. В списку подій з'являється меню додаткових команд

Додамо до першої події (натискання клавіатурної кнопки) другу подію (відпускання екранної кнопки):

```
on (keyPress "<Left>", release)
{
    prevFrame();
}
```

Повторимо процедуру для інших кнопок і в результаті одержимо фотоальбом, у якому гортання фотографій буде відбуватися як за допомогою мишки, так і за допомогою клавіатури).

У розглянутому прикладі ми використали перехід за номером кадру, але цей спосіб не завжди зручний: якщо в процесі редагування фільму нумерація кадрів зміниться, то логіка може порушитися. Більш зручно використати перехід по мітці кадру. Розглянемо приклад, у якому буде потрібно не тільки перегортання альбому, але й перехід до різних розділів, тобто більше складна навігація.

Нехай альбом буде складатися з малюнків, комп'ютерної графіки й фотографій.

Першому кадру розділу «малюнки» дамо мітку Pictures, аналогічно першим кадрам інших розділів привласнимо мітки graphics і photo.

Створимо шар для розміщення міток і назвемо його Lables. Для того щоб поставити мітку кадру, у панелі Properties необхідно

вибрати тип мітки Name і записати її ім'я. У нашому випадку Pictures. Аналогічно розмістимо мітки в кадрах 5 і 10.

Тепер додаємо новий шар і назвіть його Actions. У першому ключовому кадрі шару Actions викличіть панель Actions Frame (шляхом натискання кнопки F9) і наберіть команду stop ( ).

Додаємо ще один шар під ім'ям Subjects (тематика) у якому заголовок відповідних кадрів: «Малюнки», «Графіка» і «Фото».

Кнопки меню з аналогічними іменами розташовуємо на новому шарі за назвою Menu.

Друкуємо з лівого краю перший пункт меню «Малюнки» і переводимо в кнопковий символ. За допомогою інструмента Arrow виділяємо текстовий блок «Малюнки» і виконуємо команду Modify => Convert to Symbol (цю команду можна виконати й за допомогою клавіші F8), у панелі Convert to Symbol задаємо тип символу Button і визначаємо його ім'я як pictureButton.

Створюємо чотири кадри (рис. 14) для кнопки «Малюнки»: перший представляє вихідний текст, другий - той самий текст, але синього кольору, третій пропускаємо (у цьому випадку Down-кадр буде такий, як і Over-кадр), а в Hit-кадрі намалюємо прямокутну область, що визначить межі натискання кнопки.

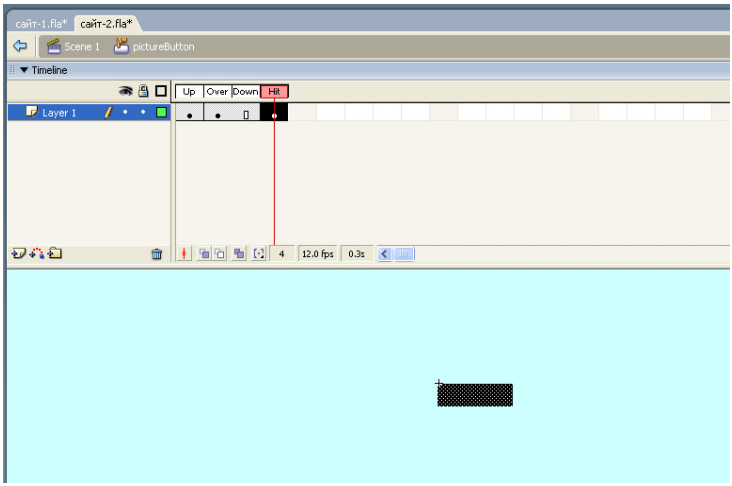


Рис. 14. Hit-кадр кнопки pictureButton.

Закріплюємо сценарій на новостворену кнопку. Для цього виділяємо кнопку в сцені Scene 1 і, натиснувши F9, викликаємо панель Actions. Внесимо необхідний код.

Аналогічно створюємо інші кнопки. Додаємо ще один шар, назваємо його Images і вносимо необхідні зображення.

Контрольні запитання:

1. Що таке інтерактивність?
2. Для чого використовуються кнопки.
3. В чому полягає автоматизоване написання скриптів?
4. В який випадках варто вручну набирати програмний код?
5. Як реалізувати автоматизацію написання скриптів?
6. Що робить такий скрипт?

```
on (release)
{
    prevFrame();
}
```

7. Що робить такий скрипт?

```
on (release)
{
    gotoAndStop(5);
}
```

Лабораторна робота № 2.4

Дослідження алгоритмів розробки ігор

Мета: створення анімацію з використанням ActionScript.

Завдання:

1. **С**творити анімований ролик про постріли гармат по мішені, використовуючи ActionScript.
2. Створити просту анімацію з оброблювачами подій, згідно із індивідуальним варіантом.

Технічне і програмне забезпечення:

1. Персональний комп'ютер з тактовою частотою процесора 2Ггц і вище.
2. Операційна система Windows 7 і новіша.
3. Програмне середовище (по вибору здобувача): NanoFL, Lightspark, Ruffle, BlueMaxima's Flashpoint, HTML5.

Основні теоретичні відомості:

Рекомендації до виконання завдання:

1. Завантажіть відповідний програмний продукт на вашому комп'ютері. Створіть новий проект **(File > New)**, який збережіть з будь-яким ім'ям (наприклад, «StarWars») і налаштуйте параметри ролику **(Modify > Movie).** Тут треба лише змінити кількість кадрів за секунду - зробимо його рівним 60.
2. Налаштуйте параметри публікації **(File > Publish Settings);** приберіть галочку з HTML і поставте галочку на **WindowsProjector**. Це означає, що ваша програма буде публікуватися не в HTML, а у вигляді exe-файлу, що виконується.
3. Створіть новий символ **(Insert > New Symbol)**, який назвіть plate (тарілка). Клацніть OK і намалюйте тарілку. Тарілка (як і всі інші символи) повинна перебувати приблизно по центру вікна (центр символу збігається із хрестиком). Щоб побачити всі символи, що знаходяться у ролику, відкрийте бібліотеку символів **(Window > Library).**
4. У такий же спосіб створіть гармату (витягнутий по горизонталі прямокутник), назвіть її cannon. Дуло для

гармати (назвіть символ **cannon_barrel**) і для тарілки **(plate_barrel)** відобразіть як вертикальні прямокутники. Не забувайте: для кожного із цих ігрових об'єктів ми створюємо новий символ **(Insert > New Symbol).** Потім створіть снаряди, якими буде стріляти тарілка й гармата (це теж окремі символи) і назвіть їх відповідно: **plate_shell і cannon_shell**. Намалюйте їх у вигляді невеликих прямокутників або кіл.

5. Після цього перемістіть по одному символу на нашу сцену (простим перетягненням символів з вікна бібліотеки). Символи, що перебувають в основному ролику, називаються екземплярами **(instance)**. В одному ролику може бути кілька екземплярів того самого символу. Цим екземплярам потрібно дати імена (ті самі, що й самим символам).
6. Розставте об’єкти так, як це потрібно (тарілка – вгорі, гармата – знизу).
7. Перейдіть до **ActionScript**. Вони бувають двох типів – **сценарії об’єкту** й **сценарії кадру**. **Сценарії об’єкту** зорієнтовані на зовнішні команди, що надходять від користувача (натискання клавіші, переміщення мишки тощо). С**ценарії кадру** можуть виконуватися й самі по собі. У даній лабораторній роботі будуть використовуватися **сценарії об’єкту**. Сама програма буде побудована в одному шарі та одному ключовому **(keyframe)** кадрі.
8. Відкрийте вікно сценаріїв **(Window > Actions)** і клацніть на першому об’єкті. Для зручності перейдіть у режим експерта **(Expert Mode)**, натиснувши **Ctrl+E** або за допомогою стрілки у верхньому правому куті. Для руху змініть x-координату (а для снарядів ще й y-координату) екземплярів об’єктів.
9. Для цього напишіть сценарій на подію натискання клавіші. Отже, натисніть на екземпляр тарілки, а потім знову на вікно **Action.** Перейдіть у режим експерта й напишіть код:

```
onClipEvent (keyDown)
{
    if (key.isdown(key.right))
    {
        _x=_x+3;
    }
    if (key.isdown(key.left))
```

```
        {
                _x=_x-3;
        }
        if (_x<10) _x=560;
        if (_x>560) _x=10;
}
```

onClipEvent – ключове слово, що означає подія.

isDown – це метод вбудованого об'єкта **key**, що повертає значення **true** (істина), якщо натиснуто зазначену клавішу. Тепер, при натисканні будь-якої клавіші, буде виконуватися код, написаний у зовнішніх дужках, у якому перевіряється, чи натиснута клавіша.

10. Натисніть на екземпляр гармати й введіть майже такий самий код:

```
onClipEvent (keyDown)
{
        if (key.isDown(ord("D")))
        {
                _x=_x+3;
        }
        if (key.isDown(ord("A")))
        {
                _x=_x-3;
        }
        if (_x<10) _x=560;
        if (_x>560) _x=10;
}
```

Для гармати клавіша **A** буде означати рух вліво, а клавіша **D** – вправо, а тарілку рухайте стрілочками.

11. Запрограмуйте дула тарілки та гармати. Спершу, натисніть на екземпляр дула тарілки, а потім знову на вікно **Action.** Перейдіть у режим експерта й напишіть код:

```
onClipEvent (keyDown)
{
if (key.isDown(key.UP))
{
_rotation = _rotation+3;
}
```

```
if (key.isDown(key.Down))
{
_rotation = _rotation-3;
}
if (_rotation<-90) _rotation=-90;
if (_rotation>90) _rotation=90;
}
onClipEvent (keyDown)
{
if (key.isdown(key.right))
{
_x=_x+3;
}
if (key.isdown(key.left))
{
_x=_x-3;
}
if (_x<10) _x=560;
if (_x>560) _x=10;
}
```

12. Аналогічно, натисніть на екземпляр дула гармати, а потім знову на вікно **Action.** Перейдіть у режим експерта й напишіть код:

```
onClipEvent (keyDown)
{
if (key.isDown(ord("W")))
{
_rotation = _rotation+3;
}
if (key.isDown(ord("S")))
{
_rotation = _rotation-3;
}
if (_rotation<-90) _rotation = -90;
if (_rotation>90) _rotation = 90;
}
onClipEvent (keyDown)
{
```

```
        if (key.isDown(ord("D")))
        {
                _x=_x+3;
        }
        if (key.isDown(ord("A")))
        {
                _x=_x-3;
        }
        if (_x<10) _x=560;
        if (_x>560) _x=10;
}
```

13. Запрограмуйте польоти снарядів. Натисніть на екземпляр снаряду тарілки, а потім знову на вікно **Action.** Перейдіть у режим експерта та напишіть код:

```
onClipEvent (keyDown)
{
if (key.isDown(key.SPACE))
{
PS = Math.sin(Math.pi*_rotation/180);
PC = Math.cos(Math.pi*_rotation/180);
_x = _x-x-40*PS;
_y = _y+40*PC;
}
else
{
flag_x = _x;
flag_y = _y;
}
}

onClipEvent (keyDown)
{
if (key.isDown(key.UP))
{
_rotation = _rotation+3;
}
if (key.isDown(key.Down))
{
```

```
_rotation = _rotation-3;
}
if (_rotation<-90) _rotation=-90;
if (_rotation>90) _rotation=90;
}
onClipEvent (keyDown)
{
if (key.isdown(key.right))
{
_x=_x+3;
}
if (key.isdown(key.left))
{
_x=_x-3;
}
if (_x<10) _x=560;
if (_x>560) _x=10;
if (_y>300) {_x=flag_x ;_y=flag_y};
}
```

14. Аналогічно, натисніть на екземпляр снаряду гармати, а потім знову на вікно **Action.** Перейдіть у режим експерта та напишіть код:

```
onClipEvent (keyDown)
{
if (key.isDown(key.SPACE))
{
CS = Math.sin(Math.pi*_rotation/180);
CC = Math.cos(Math.pi*_rotation/180);
_x = _x+40*CS;
_y = _y-y-40*CC;
}
else
{
flag_x = _x;
flag_y = _y;
}
}
onClipEvent (keyDown)
```

```
{
if (key.isDown(ord("W")))
{
_rotation = _rotation+3;
}
if (key.isDown(ord("S")))
{
_rotation = _rotation-3;
}
if (_rotation<-90) _rotation = -90;
if (_rotation>90) _rotation = 90;
}
onClipEvent (keyDown)
{
        if (key.isDown(ord("D")))
        {
                _x=_x+3;
        }
        if (key.isDown(ord("A")))
        {
                _x=_x-3;
        }
        if (_x<10) _x=560;
        if (_x>560) _x=10;
        if (_y<10) {_x=flag_x ;_y=flag_y};
}
```

15. Програма готова і її можна випробувати. Натисніть **F12** і перевірте.

Порядок виконання роботи:

1. Оберіть тему.
2. Вивчіть документацію та обробіть матеріал.
3. Виконайте роботу.
4. Оформіть звіт.
5. Завершивши роботу, здайте електронний варіант звіту через мережу викладачеві і закрийте всі вікна.
6. Передайте роздрукований примірник звіту викладачеві та захистіть роботу не пізніше закінчення наступного лабораторного заняття.

Контрольні запитання:

1. Для чого використовується ActionScript?
2. Які існують типи програмних об’єктів та чим вони відрізняються?
3. Як описується змінна?
4. Навіщо використовуються оброблювачі подій?
5. Для чого використовується адресація?
6. Які існують види адресації та чим вони відрізняються?
7. Яка мінімальна кількість кадрів необхідна для моделювання руху рота?
8. Яким рівнянням було описано рух метелика в даній роботі?
9. Яка мінімальна кількість кадрів необхідна для того, щоб метелик махав крильми?
10. Як описується зсув початку координат по осі Y?
11. Як задається крок переміщення по осі X?
12. Наведіть та поясніть рівняння кола.
13. Наведіть та поясніть рівняння коливань.
14. Назвіть основні оператори мови ActionScript.

МОДУЛЬ 3 «ІНСТРУМЕНТАЛЬНІ ЗАСОБИ ПІДГОТОВКИ ТА ВИРОБНИЦТВА ЕЛЕКТРОННИХ МУЛЬТИМЕДІЙНИХ ВИДАНЬ»

Лабораторна робота 3.1

Дослідження алгоритмів моделювання руху елементів

Мета: навчитися створювати анімацію з використанням ActionScript.

Завдання:

1. Створити просту анімацію обличчя з оброблювачами подій.

Технічне і програмне забезпечення:

1. Персональний комп'ютер з тактовою частотою процесора 2Ггц і вище.
2. Операційна система Windows 7 і новіша.
3. Програмне середовище (по вибору здобувача): NanoFL, Lightspark, Ruffle, BlueMaxima's Flashpoint, HTML5.

Основні теоретичні відомості:

Рекомендації до виконання завдання 1:

Для призначення сценарію екземпляру мувікліпу можна використати оброблювач подій кліпів **onClipEvent ()**, що здатний реагувати на різні події, наприклад, на подію появи нового кадру **enterFrame**.

Розглянемо найпростіший приклад. Створіть на сцені мувікліп у формі еліпса і "повісьте" на нього такий код.

```
onClipEvent (enterFrame)
{
        _x += 5;
}
```

З кожним новим кадром мувікліп переміщуйте на п'ять пікселів праворуч - доки не "виїде" за межі екрана, зправа.

Подія **enterFrame** генерується із частотою зміни кадрів на монтажній лінійці.

Якщо ми задамо більшу швидкість зміни кадрів, то мувікліп буде переміщуватися швидше. Крім події **enterFrame** в

оброблювачі подій кліпів **onClipEvent ()** можна, наприклад, використати подію **mouseMove**, що генерується при переміщенні курсору мишки в межах вікна FlashPlayer.

Замініть події **enterFrame** на **mouseMove**. Тепер еліпс буде рухатися тільки при переміщенні мишки в межах вікна FlashPlayer.

Створіть два мувікліпу, один із яких буде рухатися при переміщенні мишки, інший - при натисканні лівої кнопки мишки.

Відповідно, на перший мувікліп “повісьте” код:

```
onClipEvent (mouseMove)
{
        this._x+=5;
}
```

А на другий:

```
onClipEvent (mouseDown)
{
        this._x+=5;
}
```

Отримайте ролик, що показаний на рис. 4. Спробуйте, маніпулюючи мишкою, зробити так, щоб жовтий еліпс виявився під синім до того, як вони обидва не зникнуть за межею екрана зправа.

Розгляньте ще кілька прикладів, які допоможуть розібратися з відносною й абсолютною адресацією мувікліпів.

Рекомендації до виконання завдання 2:

Відносна й абсолютна адресація

Мувікліп, розташований на основній монтажній лінійці всередині може містити вкладений мувікліп. Іноді виникає необхідність звертатися (посилатися) до об'єктів, які розташовані на інших монтажних лінійках. Розгляньте це на конкретному прикладі. Складіть з еліпсів зображення чоловічка.

Створіть три мувікліпи: **eye_l** (ліве око), **eye_r** (праве око), **mouth** (рот) (рис. 15).

На їхній базі створіть мувікліп **face**, до якого будуть входити примірники цих трьох мувіклипів. Їхні імена визначимо відповідно як **mouth1, eye1, eye2** (рис. 15). Створіть примірник мувікліпу **face** на сцені й дайте йому ім'я **face1**.

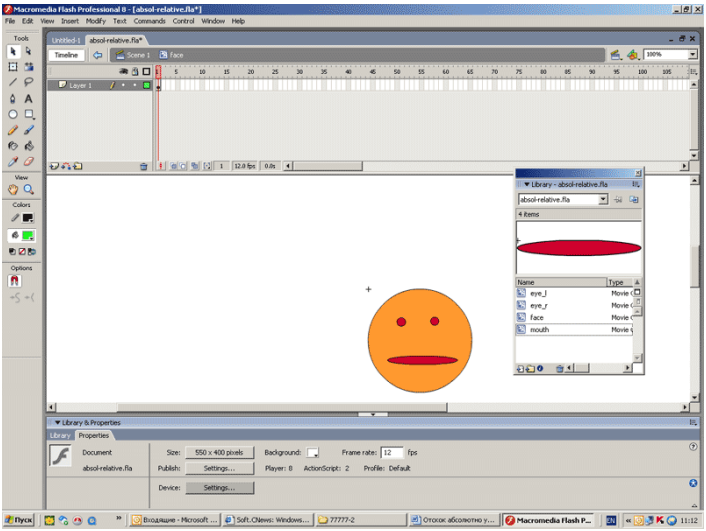


Рис. 15. Демонстрація створення зображення.

Тепер задайте нескладну анімацію. Наприклад, при натисканні на кнопку мишки чоловічок відкриває рота. Ця дія буде задаватися масштабуванням примірника мувікліпу **mouth1** по вертикалі.

Отже, розгляньте, які є варіанти звертання до примірника мувікліпу **mouth1**.

Використайте спочатку абсолютну адресацію.

Абсолютна адресація припускає вказівку шляху із самого верхнього рівня – з основної монтажної лінійки. Для посилання на неї використовується властивість **_root.**

Редактор ActionScript дозволяє автоматично визначити шлях до об'єкта.

Запишіть рядок коду в панелі Actions.

```
onClipEvent (mouseDown)
{

}
```

Потім натисніть на значок мішені (**Insert a Target Path**) (див. рис. 7). У результаті з'явиться однойменна панель. Натисніть на хрестик у віконці панелі й побачте структуру вкладеності (рис. 8).

Виділіть потрібний елемент (**mouth1)** і виберіть тип адресації **Absolute** (абсолютний).

При натисканні кнопки OK потрібний вираз автоматично з'явиться в рядку коду.

Далі визначіть збільшення масштабу мувікліпу **mouth1** по осі у на 170%.

```
onClipEvent (mouseDown)
{
        _root.face1.mouth1._yscale = 170;
}
```

Відповідно при виборі відносної адресації матимемо інший, але теж коректний вираз.

```
onClipEvent (mouseDown)
{
        this.mouth1._yscale = 170;
}
```

Рис. 11 ілюструє роботу даного коду. Дещо ускладнимо приклад: нехай рот анімації розкривається в кожному новому кадрі (тобто масштаб мувікліпу **mouth1** по осі у збільшується на 20% у кожному новому кадрі), а по досягненні 150% - скорочується до 20%.

```
onClipEvent (enterFrame)
{
this.mouth1._yscale += 20;
if (this.mouth1._yscale > 150)
this.mouth1._yscale =20;
}
```

У результаті отримаємо ролик із чоловічком, що відкриває/закриває рот. У цьому прикладі дещо змінили положення мувікліпу **mouth** щодо центра, щоб масштабування еліпса більше нагадувало ефект відкривання/закривання рота.

Щоб розглянути більш складні приклади, знадобиться введення поняття змінних, з метою правильного їх оголошення.

Змінна - це контейнер, що містить інформацію. Ім'я змінної в даному скрипті повинне бути постійним, а значення її може змінюватися. На одній тимчасовій шкалі, в одному кліпі або кнопці не повинно бути двох змінних з однаковими іменами.

Змінна складається з літер латинського алфавіту, нижнього підкреслення, значка $ і цифр, і не повинна починатися з цифр.

Присвоєння початкового значення змінній називають ініціалізацією змінної.

var myVariable:Number = 5;

Дана змінна містить числове значення 5. Вираз **«:Number»** вказує на те, що тип даних числовий.

Одночасно можна визначити декілька змінних, наприклад:

var a=3, b=4, c=5;//визначення декількох змінних

Однак запис типу **myVariable=5;**

теж є коректним. Якщо в сценарії здійснюється присвоєння значення змінної, котра попередньо не була оголошена, то інтерпретатор автоматично створить змінну з типом значення, що їй привласнюється. Тут ми будемо говорити лише про змінні числового типу.

Зазначимо, що в ActionScript текст, написаний після знака операнду (//), сприймається як коментар, що ігнорується плеєром. Коментарі допоможуть читати текст вашої програми.

Для того, щоб перевірити значення змінної можна звернутися до функції **trace**, що має наступний синтаксис: trace (expression:Object).

Такий код ілюструє роботу цієї функції:

Number = 2;

trace(Number) //виводить значення 2

Отже, після того як частково ознайомилися із змінними, поверніться до прикладу 1 і дещо ускладніть його - нехай еліпс робить зворотно-поступальні рухи, змінюючи координату по осі **X** у межах від 10 до 300 пікселів.

Такий код можна записати, ввівши змінну **k1**, що буде використовуватися як коефіцієнт і змінює знак збільшення координати по осі **X**. Значення змінної **k1** буде визначено поза кодом, що додається до кліпу і змінюється всередині коду кліпу.

Зверніть увагу: якщо задати змінну **k1** у кадрі, в якому розташований мувікліп як **k1=1**, то програма працювати не буде. Подібне завдання значення змінної є некоректним. Використовуючи функції trace у панелі output, матимемо значення цієї змінної – **undefined**. Отже, в області видимості сценарію мувіклипа змінна **k1** не визначена. Для того, щоб звернутися до цієї змінної зі сценарію мувікліпу, необхідно дати йому ім'я, наприклад **mc1** і використати крапковий синтаксис. Тобто в кадрі основної монтажної лінійки записати **mc1.k1=1**.

У цьому випадку все працює коректно. Використовуючи функцію **trace**, можна побачити, як значення змінної **k1** змінюється на протилежне, коли еліпс відбивається від межі.

```
onClipEvent (enterFrame)
{
        trace( k1);
        _x += 5 * k1;
        if (_x>300) k1 = -1;
        if (_x<10) k1 = 1;
}
```

У результаті отримаємо ролик-приклад.

Доречним вважається запис коду в одному місці, оскільки в цьому випадку його простіше читати.

Більш компактний код можна отримати, використовуючи системний оброблювач події появи нового кадру **onEnterFrame**. Для того, щоб при виникненні певної події виклик методу оброблювача зініціював певні дії, він повинен мати відповідні функції.

У записі **_root.onEnterFrame = function (){}** об'єкт **_root.** (основна тимчасова шкала) слухає подію **onEnterFrame** (початок нового кадру), і з початком цієї події (новий кадр) виконується функція **function (){}**. Розглянемо приклад, у якому функція буде визначати зміну координат певного мувікліпу, розташованого на основній тимчасовій шкалі. Зміни координат мувікліпу будуть вироблятися із частотою зміни кадрів на основній монтажній лінійці. Зокрема, використовуючи такий код, записаний у кадрі основної сцени:

```
k1=1
_root.onEnterFrame = function (){
trace( k1);
mc1._x += 5 * k1;
if (mc1._x>300) k1 = -1;
if (mc1._x<10) k1 = 1;
}
```

отримаємо анімований ролик, аналогічний попередньому.

Ускладнимо траєкторію руху мувікліпу, наприклад, задамо рух по синусоїді.

Для того щоб використати математичні формули, зверніться до методів або констант об'єкта **Math**.

Для використання цього об'єкта в палітрі Actions зверніться до меню **Objects (**об'єкт**) > Core (**ядро**) > Math** (математика)**.**

Використовуючи об'єкт **Math,** застосовуєте такий синтаксис: **Math.method(parameter)** або **Math.constant.**

Наприклад, число запишіть як Math.PI.

Якщо необхідно "витягти" корінь із певної змінної **y1** і зберегти результат обчислень у змінній **x1**, можна використати вираз:

x1 = Math.sqrt(y1).

Якщо необхідно піднести до степеня **y** певну змінну **x**, можна використати вираз:

Math.pow(x, y);

Наприклад:

trace(Math.pow(2, 10)); // Дає результат: 1024

Порядок виконання роботи:

1. Обгрунтуйте тему у відповідності до варіанту.
2. Вивчіть документацію та обробіть матеріал.
3. Виконайте роботу.
4. Оформіть звіт.
5. Завершивши роботу, здайте електронний варіант звіту через мережу викладачеві і закрийте всі вікна.
6. Передайте роздрукований примірник звіту викладачеві та захистіть роботу не пізніше закінчення наступного лабораторного заняття.

Варіанти індивідуальних завдань (задає викладач).

Контрольні запитання:

1. Для чого використовується ActionScript?
2. Які існують типи програмних об'єктів та чим вони відрізняються?
3. Як описується змінна?
4. Навіщо використовуються оброблювачі подій?
5. Для чого використовується адресація?
6. Які існують види адресації та чим вони відрізняються?
7. Яка мінімальна кількість кадрів необхідна для моделювання руху рота?

Лабораторна робота 3.2

Дослідження інтерфейсу навігації для зображень та технології створення ефектів

Мета: навчитися створювати анімацію та різноманітні ефекти з використанням ActionScript.

Завдання:

1. Створити анімацію падаючого м'яча.
2. Створити анімацію маятника.

Технічне і програмне забезпечення:

1. Персональний комп'ютер з тактовою частотою процесора 2Ггц і вище.
2. Операційна система Windows 7 і новіша.
3. Програмне середовище (по вибору здобувача): NanoFL, Lightspark, Ruffle, BlueMaxima's Flashpoint, HTML5.

Основні теоретичні відомості:

Рекомендації до виконання завдання 1:

Рух м'яча, що падає з висоти H, опишіть рівнянням

$$y = H - gt2/2$$

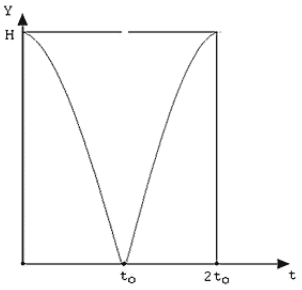


Рис. 16. Траєкторія руху м'яча на папері.

Нехай через час t_0 кулька досягне поверхні (рис 16), тоді t_0 можна визначити як корінь квадратний з $2H/g$.

На ділянці за часом від **0** до $\mathbf{t_0}$ координату мувікліпу **mc1** по осі **y** можна обчислити по формулі:

mc1._y = (H -g* t*t*0.5);

На ділянці за часом від **t0** до **2 t0** координату по осі **y** обчисліть за формулою:

mc1._y = (H -g* (2*t0* -t)*(2*t0* -t) *0.5);

Для того щоб користуватися однією формулою, введіть співмножник **n** (параметр, що визначає фазу руху (падіння (n = 0) або відскік (n = 1)), що буде дорівнювати нулю при обчисленні на ділянці за часом від **0** до $\mathbf{t_0}$ і одиниці при обчисленні на ділянці за часом від $\mathbf{t_0}$ до $\mathbf{2t_0}$.

У цьому випадку вираз матиме вигляд:

mc1._y = (H -g* (2*t0*n -t)*(2*t0*n -t) *0.5);

З огляду на те, що в Flash координата по осі **y** спрямована зверху вниз, для того, щоб кулька падала зверху вниз, а не навпаки, введіть параметр **y1** і перепишіть вираз з урахуванням зміни напрямку координати **y**.

mc1._y = y1 - (H -g* (2*t0*n -t)*(2*t0*n -t) *0.5);

Оскільки ми моделюємо відскік абсолютно пружного м'яча й зневажаємо будь-якими втратами на тертя об повітря й т.п., то потрібно задати лише один цикл руху м'яча - всі інші будуть такими ж. Після визначення циклу за часом запишіть умову:

if (t > 2* t0) t=dt;

Тобто, для всіх значень часу більше одного циклу руху м'яча (падіння - відскік) відлік часу починаємо з початку.

У підсумку рух м'яча можна описати наступним кодом:

```
y1=250;     // зрушення початку координат по осі y
H=200;      // висота, з якої падає м'яч
x1=600;     // зрушення початку координат по осі x
g=9.8;      // прискорення вільного падіння
t=0;        // початкове значення часу
n=1;        // фаза руху (відскік або падіння)
dt= 0.2;    // крок за часом
k=1;
dx = 2;
_root.onEnterFrame = function ()
{
      t = t+ dt;
      t0 = Math.sqrt(2* H/g);
```

```
            if (t > 2* t0) t=dt;
            if(t < t0 ) n=0;
            if (t>= t0) n=1;
            mc1._y = y1- (H -g* (2*t0*n -t)*(2*t0*n -t) *0.5);
            trace (t);
            trace (mc1._y);
        }
```

У результаті отримаємо готовий анімований ролик.

Рекомендації до виконання завдання 2:

Пригадаємо основні кінематичні характеристики коливань:

- амплітуда коливань (А) - максимальна відстань, на яку віддаляється коливне тіло від свого положення рівноваги. Амплітуда вільних коливань визначається початковими умовами (м);
- період коливання (Т) - це мінімальний проміжок часу, після закінчення якого система повертається в колишній стан; інакше кажучи, період коливання - це час, за яке відбувається одне повне коливання;
- частота коливань (ν) - це число коливань за 1 с, виміряється в герцах (Гц);
- циклічна частота (ω) - це величина, в 2π разів більша частоти. Вона показує, яке число коливань відбувається за 2π секунд.

Період, частота й циклічна частота пов'язані виразами:

$$\omega = 2\pi\upsilon$$

$$\upsilon = \frac{n}{t} = \frac{1}{T}; \quad T = \frac{t}{n} = \frac{1}{\upsilon}; \quad T = \frac{2\pi}{\omega}$$

де *n* - число коливань, а *t* - час, за який відбулося *n* коливань.

Якщо тертя настільки незначне, що ним можна знехтувати, то графіком залежності координати коливного тіла (матеріальної крапки) від часу є синусоїда.

Якщо момент початку відліку часу коливань збігається з моментом максимального відхилення маятника від положення рівноваги, рівняння коливань буде таким:

$$x = A\sin\alpha$$

або ,

$$x = A\sin\omega t$$

тобто коливання будуть синусоїдальними й відбуватися без початкової фази; **x** - зсув маятника (рис 17).

Якщо момент початку відліку часу коливань не збігається ні з моментом максимального відхилення від положення рівноваги, ні з моментом проходження ним положення рівноваги, то коливання відбуваються з початковою фазою й рівняння таких коливань має такий вид:

$$x = A\cos(\omega t + \alpha_0)$$

або .

$$x = A\sin(\omega t + \alpha_0)$$

Фаза коливань - це величина, що дозволяє визначити, яка частина періоду пройшла з моменту початку коливань і найбільш повно характеризує коливальний процес:

$$\alpha = \omega t + \alpha_0$$

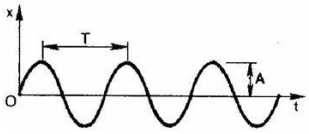


Рис. 17. Графік коливання матеатичного маятника.

Математичний маятник - це ідеалізована модель, що правильно описує існуючий маятник лише за певних умов. Реальний маятник можна вважати математичним, якщо довжина нитки набагато більша розмірів підвішеного на ній тіла. Маса нитки мізерно мала в порівнянні з масою тіла, а деформації нитки настільки малі, що ними можна знехтувати.

Період вільних коливань математичного маятника не залежить від його маси, а визначається лише довжиною нитки й прискоренням вільного падіння в тім місці, де перебуває маятник.

$$T = 2\pi\sqrt{\frac{l}{g}}.$$

Рух маятника

Отже, перейдемо до створення відео-ролика. Створіть екземпляр мувікліпу Symbol1 і дайте йому ім'я **mc1**, а для того, щоб змусити його рухатися, напишіть код.

Спочатку введіть такі змінні:

ampl – амплітуда коливань маятника (м);

l – довжина нитки маятника (м);

g – прискорення вільного падіння (м/с2);

Lx – зрушення початку координат по осі X (у пікселях);

Ly – зрушення початку координат по осі Y (у пікселях);

K – коефіцієнт переказу переміщень (м) у піселях;

dt– крок за часом (с).

Період коливань маятника обчисліть за формулою

T = 2 * Math.PI / Math.sqrt(l/g);

Вираз **x=A Sin(t/T2)** застосуйте для обчислення зміни координати **x** мувікліпу **mc1**, що прийме вигляд:

mc1._x = x* K + Lx;

де

x = ampl * Math.sin(t/T * 2* Math.PI);

Координата **y** повинна змінюватися за рівнянням кола:

x2 + y2 = R2;

З огляду на те, що радіус дорівнює довжині нитки l, а напрямок координати **y** в Flash спрямовано зверху вниз, та зміну координати **y** мувікліпу **mc1** можна знайти за виразом:

mc1._y = y*K – Ly; де **y = Math.sqrt(l*l - x*x)+l;**

У результаті рух маятника можна описати таким кодом:

```
ampl=0.5;      // амплітуда коливань маятника
l = 1;         // довжина нитки маятника
g=9.8;         // прискорення вільного падіння
Lx = 400;      // зрушення початку координат по осі x
Ly =600;       // зрушення початку координат по осі y
K = 500;       // коефіцієнт переміщень маятника (у пікселях)
t=0;           // початкове значення часу
dt = 0.05;     // крок за часом
T= 2 * Math.PI / Math.sqrt(l/g); //період коливань маятника
trace ("T" +"="+ T);
_root.onEnterFrame = function ()
{
t = t + dt;
x1 = ampl * Math.sin(t/T * 2* Math.PI);
y1 = Math.sqrt(l*l - x1*x1)+l;
mc1._y = y1*K - Ly;
mc1._x = x1* K + Lx;
trace ("mc1._x" +"="+ mc1._x);
trace ("mc1._y" +"="+ mc1._y);
}
```

У результаті отримаємо готовий анімований ролик.

Порядок виконання роботи:

1. Обгрунтуйте тему у відповідності до варіанту.
2. Вивчіть документацію та обробіть матеріал.
3. Виконайте роботу.
4. Оформіть звіт.
5. Завершивши роботу, здайте електронний варіант звіту через мережу викладачеві і закрийте всі вікна.
6. Передайте роздрукований примірник звіту викладачеві та захистіть роботу не пізніше закінчення наступного лабораторного заняття.

Варіанти індивідуальних завдань (задає викладач).

Контрольні запитання:

1. Яким рівнянням описується рух м’яча, що падає з висоти H?
2. Яка мінімальна кількість кадрів необхідна для моделювання руху маятника?
3. Яким рівнянням описується рух маятника?
4. Чим відрізняється математичний маятник від фізичного?
5. Як описується зсув початку координат по осі Y?
6. Як задається крок переміщення по осі X?
7. Наведіть та поясніть рівняння кола.

Лабораторна робота 3.3

Дослідження інтерфейсу компоненти ActionScript

Мета: навчитися створювати анімацію та різноманітні ефекти з використанням ActionScript.

Завдання:

Створити анімацію польоту метелика.

Технічне і програмне забезпечення:

1. Персональний комп'ютер з тактовою частотою процесора 2Ггц і вище.
2. Операційна система Windows 7 і новіша.
3. Програмне середовище (по вибору здобувача): NanoFL, Lightspark, Ruffle, BlueMaxima's Flashpoint, HTML5.

Основні теоретичні відомості:

У цьому випадку мова йде не про те, щоб точно зімітувати фізику польоту комахи, а про створення грубої імітації, що може прикрасити, наприклад, Web-сторінку.

Розбийте завдання на два етапи: анімацію складання крил зробіть дизайнерськими засобами, а переміщення мувікліпу (метелика, що махає крильми) опишіть на основі закону псевдовипадкових чисел.

Для даного проекту використайте растрове зображення метелика, переведіть його до векторного вигляду і збережіть як мувікліп (рис. 18).

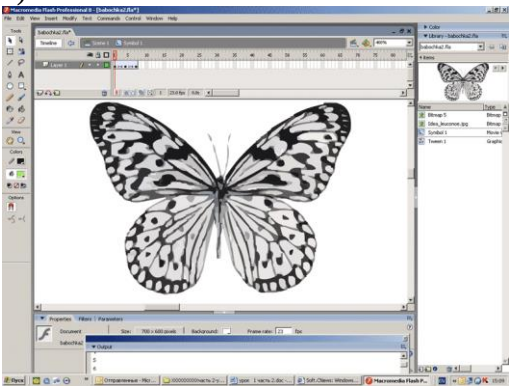


Рис. 18. Завантаження растрового зображення метелика.

Потім задайте анімацію Motion Tween (три кадри на розкриття крил, один кадр - повністю складені крила й три кадри на розкриття - усього сім кадрів.

Після цього створіть екземпляр мувікліпу на шарі layer1 і привласніть йому ім'я **mc1**.

На іншому шарі (який названий layer2) помістіть заднє тло з написом (рис. 19).

Рис. 19. Створення другого шару зображення.

Ми хотіли, щоб метелик робив змах крильми на одному місці (на це в нас іде сім кадрів), а потім переміщувався на випадкову відстань в одному із чотирьох напрямків, котрий буде обраний довільно. Крім цього, на кожному кроці метелик додатково повинен рухатися на задані збільшення вперед і вправо, а, долетівши до межі екрана, змінюти свій напрямок. Тобто в переміщенні по обох осях повинні бути дві складові - випадкова й закономірна.

Момент, коли необхідне переміщення метелика (сьомий кадр) відслідковувався за умовою

if(n == 7).

Для того, щоб випадково вибрати довільний напрямок руху, ми зводили (-1) у ступінь випадкового числа (від 0 до 10), округленого до цілого.

rn = Math.random()*10; // випадкове число від 0 до 10
irn = Math.round(rn); // випадкове число, округлене до цілого
dir_change = Math.pow((-1),irn);

де змінна **dir_change** (зміна напрямку) приймає значення 1 або (–1) випадковим чином.

Відповідно, переміщення мувікліпу **mc1** по осі **x** можна записати за допомогою виразу:

mc1._x = mc1._x + Math.random()* ksx * dir_change + m*dx;

де **Math.random()*ksx** – випадкова складова переміщення, а **m*dx** – невипадкова. Коефіцієнт **m** змінюється на протилежне значення (1 або –1) при відбитті від лівої й правої межі екрана.

Лістинг 1: Зразок написання програми польоту метелика.

```
// ************************
// *   Політ метелика     *
// ************************
x1=300;        // координата первісного положення метелика
                  на екрані по осі x
y1=400;        // координата первісного положення метелика
                  на екрані по осі y
Rb= 600;       // права межа
Lb= 10;        // ліва межа
Bb = 500;      // нижня межа
Ub= 10;        // верхня межа
n=0;           // лічильник кількості змахів крильми
k=1;           //  змінна зміни напрямку руху по осі y
m=1;           // змінна зміни напрямку руху по осі y
ksx = 20;      // коефіцієнт випадкового зсуву по осі x
ksy = 30;      // коефіцієнт випадкового зсуву по осі y
dx = 10;       // додаткове (невипадкове) зсув по осі x
dy =10;        // додаткове (невипадкове) зсув по осі y
// первісне положення метелика на екрані
mc1._x = x1;
mc1._y = y1;
_root.onEnterFrame = function ()
{
      n=n+1;
      trace (n);
```

```
        rn = Math.random()*10; // випадкове число від 0 до 10
        irn = Math.round(rn);     // випадкове число, округлене
                                  до цілого
        dir_change = Math.pow((-1),irn);
        // Переміщення метелика відбувається після повного
          циклу змахів крил
        if(n == 7)
        {
                mc1._x = mc1._x + Math.random()* ksx *
                        dir_change + m*dx;
                mc1._y = mc1._y + Math.random()* ksy *
                        dir_change + k*dy;
                // умова відбиття від межі екрана
                if (mc1._x > Rb ) m=-1;
                if (mc1._x < Lb ) m=1;
                if (mc1._y > Bb ) k= -1;
                if (mc1._y < Ub ) k=1;
                //trace ("mc1._y" +"="+ mc1._y);
                //trace(k);
        }
        if(n > 6) n=0;
}
```

Порядок виконання роботи (задає викладач).

Контрольні запитання:

1. Яким рівнянням було описано рух метелика в даній роботі?
2. Яка мінімальна кількість кадрів необхідна для того, щоб метелик махав крильми?
3. Як описується зсув початку координат по осі Y?
4. Як задається крок переміщення по осі X?
5. Наведіть та поясніть рівняння кола.
6. Наведіть та поясніть рівняння коливань.
7. Назвіть основні оператори мови ActionScript.

Лабораторна робота 3.4

Дослідження технології створення 3D-графіки

Мета: навчитися створювати 3D-графіку на Unity.

Завдання:

51 Створити просту сцену з зображенням ландшафту місцевості (рис. 20).

Технічне і програмне забезпечення:

1. Персональний комп'ютер з тактовою частотою процесора 2Ггц і вище.
2. Операційна система Windows 7 і новіша.
3. Програмне середовище Unity3D 5 і новіше.

Основні теоретичні відомості:

Після запуску середовища переходимо на вкладку **Create New Project** та відмічаємо галочками набори стандартних об'єктів та скриптів, які нам знадобляться.

- Character Controller
- Particles
- Physic materials
- Scripts
- Skyboxes
- Terrain Assets
- Tree Creator

Розглянемо освовні вікна середовищаUnity 3D.

1. Інспектор префабів та ресурсів – це місце, де відображаються додані до проекта моделі, текстури, звуки, й відповідно, префаби (збережені для повторного використання об'єкти). Припустімо, що при розробці 3D-гри ми створили об'єкт супротивника з прикріпленою до нього моделлю та скриптом, що відповідає за його поведінку, та хочемо, щоб на кожному рівні не довелося створювати його заново, та щоб всі копії цього об'кту змінювались не вручну (кожна окремо), а всі разом. В цьому випадку ми зберігаємо даний об'єкт як префаб, й коли він нам знадобиться ми просто пересунемо його

мишкоб на сцену, а при зміні його, відповідно, зміняться й усі його копії..

2. Ієрархія об’єктів на сцені – це список всіх об’ктів на поточному рівні, де видно також всі відношення між об’ктами, на кшталт Parent-Child.
3. Інспектор об’єктів відображає компоненти та властивості виділеного в даний момент об’єкта (моделі, текстури, префаба).
4. Чарівні клавіші дозволяють прямо в редакторі запустити рендеринт сцени, перевірити її на наєвність багів, поставити на паузу, щоб перевірити стан будь-яких об’єктів та здійснити детальне налаштування шляхом по кадрового виконання.
5. Головне вікно редагування дозволяє розташовувати об’кти по рівням сцени. Основні клавіші для роботи у вікні редагування:
   - середня клавіша миші – переміщення камери;
   - ліва клавіша миші – виділення об’єкту;
   - права клавіша миші – обертання камери;
   - клавіша F клавіатури – центрування на виділеному об’єкті;
   - клавіша CTRL клавіатури – переміщення об’єкта на 1.

Примітка: за 1 координат краще брати 1 реальний метр. Розглянемо, які об’єкти включаються редактором до сцени при її створенні: головна камера, натиснувши на яку ми побачимо конус області виведення (viewport) у вигляді маленького віконця.

Порядок виконання роботи:

1. Оберіть тему.
2. Вивчіть документацію та обробіть матеріал.
3. Виконайте роботу.
4. Оформіть звіт.
5. Завершивши роботу, здайте електронний варіант звіту через мережу викладачеві і закрийте всі вікна.
6. Передайте роздрукований примірник звіту викладачеві та захистіть роботу не пізніше закінчення наступного лабораторного заняття.

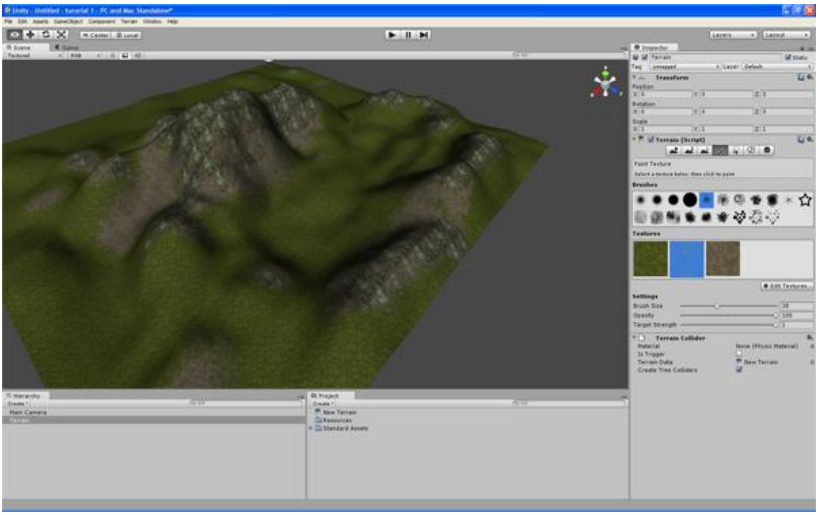


Рис. 20. Вигляд створеної сцени та вкладок при роботі з нею.

Варіанти індивідуальних завдань (задає викладач).

Контрольні запитання:

1. Які формати файлів використовуються при виконанні даної лабораторної роботи?
2. Розкажіть про пакет **Unity 3D** і його особливості.
3. Що таке 3D графіка?
4. Охарактеризуйте етапи своєї роботи.
5. Які переваги при роботі в **Unity 3D** у порівнянні з іншими аналогічними програмами?
6. Що таке примітив?
7. Що таке текстура об’єктів?
8. Розкрийте абревіатуру САПР.

Лабораторна робота 3.5

Дослідження технології створення анімації в 3D-сцені

Мета: навчитися створювати анімацію на Unity.

Завдання:

Створити просту сцену з зображенням анімованої фігурки, згідно із індивідуальним варіантом.

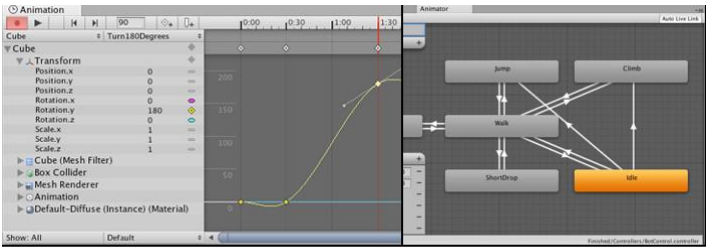
Технічне і програмне забезпечення:

1. Персональний комп'ютер з тактовою частотою процесора 2Ггц і вище.
2. Операційна система Windows 7 і новіша.
3. Програмне середовище Unity3D 5 і новіше.

Основні теоретичні відомості:

В Unity3D існує два вікна для задавання анімації (рис. 21):

1. Анімація (віддображається в вікні «Animation»)
2. Дерево анімацій (віддображається в вікні «Animator»).



Animation Animator

Рис. 21. Вигляд вікон **«Анімація»** та **«Аніматор»**.

Animation

По суті, це таймлайн з ключовими кадрами. Тут можна рухати, обертати, масштабувати будь-які об'єкти. Відповідно, можна намалювати сплайни та використовувати різноманітні зсуви. А також управляти будь-якими їх властивостями. Наприклад, створити параметр «яскравість» та змінювати його так само, як і параметри x, y, z, штатними засобами. Спрайти підтримують по кадрову анімацію (рис. 22).

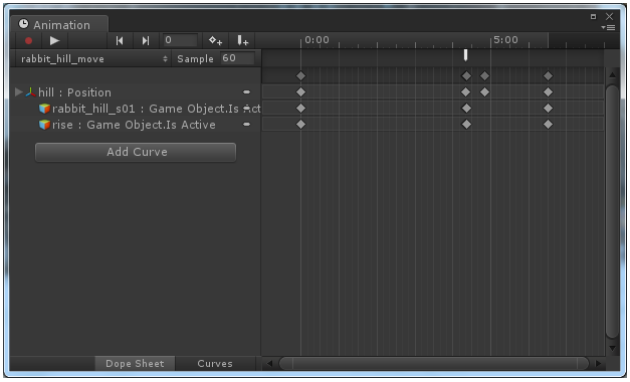


Рис. 22. Налаштування покадрової анімації.

Доречі, не зважаючи на те, що при кожній анімації є FPS (частота зміни кадрів), самі анімації до FPS не прив'язані. Вони прив'язані до часу. Тобто, якщо створити анімацію з 5 FPS, де об'єкт рухається з точки А в точку Б за допомогою задавання двох ключових кадрів (на початку та в кінці), то після рендерингу цей об'єкт буде рухатись східцями з інтервалами 5 FPS. Анімація розраховується кожний кадр гри, а FPS всередині анімації зроблено лише для зручності розробника, щоб не ставити надто часто ключові кадри.

Animator

Це більша й складніша система, яка безпосередньо керує анімаціями. Тобто анімація – це лише файл (ресурс) з налаштуваннями ключових кадрів й само по собі нічого не робить. А ось компонент «Animator» може програвати ці анімації. Крім цього можна створювати дерево таких анімацій з морфіном між ними. Тобто. Якщо є персонаж, анімований перекладками (коли кожна частина тіла є окремим спрайтом, який обертається чи рухається), то можна створити анімацію кожної части тіла окремо, на кшталт польоту метелика з попередніх робіт. А потім за допомогою мишки налаштувати умову, що залежно від швидкості руху об'єкта Animator буде вмикати анімацію ніг в режим «ходьба» чи «біг». А рухати руками персонаж буде окремою анімацією, яка не пов'язана з швидкістю переставляння ніг.

Як правило, вікно Animator буде виглядати ось так (рис. 23) та містити лише одну анімації без жодних зв'язків чи переходів.:

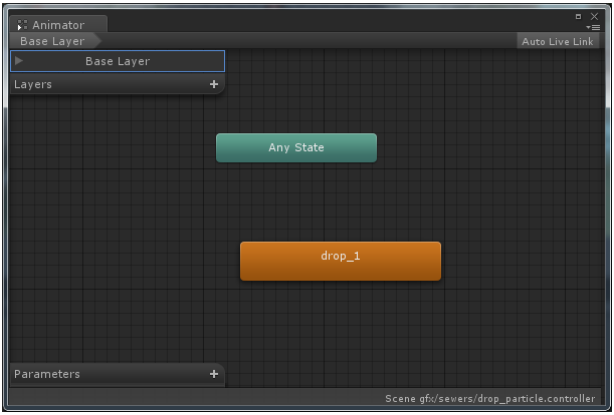


Рис. 23. Налаштування запуску анімації в вікні «**Аніматор»**.

Порядок виконання роботи:

1. Оберіть тему.
2. Вивчіть документацію та обробіть матеріал.
3. Виконайте роботу.
4. Оформіть звіт.
5. Завершивши роботу, здайте електронний варіант звіту через мережу викладачеві і закрийте всі вікна.
6. Передайте роздрукований примірник звіту викладачеві та захистіть роботу не пізніше закінчення наступного лабораторного заняття.

Варіанти індивідуальних завдань (задає викладач).

Контрольні запитання:

1. Які інструменти необхідно використати для створення анімації?
2. Розкажіть про покрокову анімацію в **Unity 3D** та її особливості.
3. Розкрийте термін “анімація”.
4. Що таке анімація?
5. Охарактеризуйте етапи своєї роботи.
6. Що таке комп’ютерна графіка?
7. Які переваги при роботі в **Unity 3D** у порівнянні з іншими аналогічними програмами при створення анімації?
8. Розкрийте абревіатуру САПР.

Лабораторна робота 3.6

Дослідження технології створення маніпулятора в Unity

Мета: навчитися створювати маніпулятори, що здійснюватимуть взаємодію з користувачем об’єктів 3D-сцени на Unity.

Завдання:

Створити маніпулятор для руху об’єкта за вимогою користувача, згідно із індивідуальним варіантом.

Технічне і програмне забезпечення:

1. Персональний комп’ютер з тактовою частотою процесора 2Ггц і вище.
2. Операційна система Windows 7 і новіша.
3. Програмне середовище Unity3D 5.0 і новіше.

Основні теоретичні відомості:

Маніпуля́тор (англ. *manipulator*) — керований пристрій (машина), оснащений робочим органом для виконання рухових функцій, аналогічних до функцій руки людини, під час переміщення об'єктів у просторі. Залежно від виду систем керування розрізняють маніпулятори з ручним і автоматичним керування.

Виконавчий механізм будь-якого маніпулятора — це багатоланковий просторовий механізм, який може мати у загальному випадку поступальні, обертальні, циліндричні, та сферичні рухи. Залежно від поставленої задачі маніпулятор повинен забезпечувати різне число ступенів свободи захоплювача. Наприклад, для відтворення просторового руху захоплювача у загальному випадку маніпулятор повинен мати шість ступенів свободи, які можна реалізувати за допомогою семиланкового кінематичного ланцюга з виключно обертовими рухами. Якщо ж потрібно відтворювати просторову траєкторію лише однієї точки захвату, то необхідне число ступенів свободи зменшується до трьох, тобто з'являються надлишкові ступені свободи. Надлишкові ступені свободи дають змогу оптимізувати кінематичні, динамічні, енергетичні та інші критерії якості процесу маніпулювання. Надлишкові (зайві) ступені свободи називають також *маневреністю маніпулятора*, яка є важливою характеристикою

маніпулятора. Збільшення числа ступенів маневреності маніпулятора розширює його можливості при виконанні складних рухів: збільшує робочий простір, зменшує мертві зони, розширює варіантність вибору траєкторій рухів у стиснених умовах.

В даній роботі необхідно навчитися створювати віртуальний маніпулятор для управління рухом об'єктів нашої віртуальної сцени з клавіатури. Для цього необхідно зробити наступні кроки:

1. Розмістити на сцені з попередньої роботи об'єкт, що хочемо рухати.
2. Виконати команду **Assets/Create/C# script** або **Assets/Create/Java script** та задати його ім'я.
3. Двічі клацнути на створеному скрипту для запуску середовища **MonoDevelop** для редагування скрипта.
4. Вввести необхідний програмний код, що допоможе здійснити необхідні переміщення (див. Лістинг 2).
5. Виконати команду **Build/Build all** для компіляції скрипта.
6. Повернутись до нашої сцени, виділити мишкою об'єкт, який хочемо рухати, та виконати команду **Components/Scripts/<Ім'я скрипта>**, щоб прив'язати скрипт до об'єкта сцени.

Лістинг 2: C#-скрипт маніпулювання об'єктом сцени з клавіатури/ миші.

```
using UnityEngine;
using System.Collections;
public class <Ім'я скрипта>: MonoBehaviour
{
	private float speed;
	private float x;
	private float y;
	private Quaternion fromRotation;
	private Quaternion toRotation;
	void Start ()
	{
		speed = 1;
	}

	void Update ()
	{
		transform.position = new Vector3 ();
		if (Input.GetMouseButton (0))
```

```
		{
			x -= Input.GetAxis ("Mouse X") * speed;
			y += Input.GetAxis ("Mouse Y") * speed;
			fromRotation = transform.rotation;
			toRotation = Quaternion.Euler (y, x, 0);
			transform.rotation = Quaternion.Lerp (fromRotation,
toRotation, Time.deltaTime * 1/speed);
		}
		if (Input.GetKey(KeyCode.LeftArrow))
		{
			transform.position += transform.right * speed;
		}
		if (Input.GetKey(KeyCode.RightArrow))
		{
			transform.position += transform.up * speed;
		}
	}
}
```

Порядок виконання роботи:

1. Оберіть тему.
2. Вивчіть документацію та обробіть матеріал.
3. Виконайте роботу.
4. Оформіть звіт.
5. Завершивши роботу, здайте електронний варіант звіту через мережу викладачеві і закрийте всі вікна.
6. Передайте роздрукований примірник звіту викладачеві та захистіть роботу не пізніше закінчення наступного лабораторного заняття.

Варіанти індивідуальних завдань (надає викладач).

Контрольні запитання:

1. Які існують види маніпуляторів?
2. Розкажіть про середовище **MonoDevelop** і його особливості.
3. Розкрийте термін “маніпулятор”.
4. Як отримати дескриптор лівої кнопки миші?
5. Охарактеризуйте етапи своєї роботи.
6. Як отримати дескриптор будь-якої клавіші на клавіатурі?
7. Які види скриптів можна додати при роботі в **Unity 3D**?
8. Як створити Touch-маніпулятор для сенсорного екрану?
9. Розкрийте абревіатуру САПР.

Рекомендована література

1. Сучасні технології електронних мультимедійних видань: монографія / Під ред.. О.І. Пушкаря. – Харьків: ВД «ІНЖЕК», 2011. - 296 с.
2. Киричок Т.Ю. Електронні видання. – К.: НТУУ «КПІ», 2010. – 400 с.
3. Мультимедійні системи як засіб інтерактивного навчання: Посібник / авт.: Жалдак М.І., Шут М.І. та ін. / За ред.: Жука Ю.О. – К.: Педагогічна думка, 2012. – 112 с.
4. Дурняк Б.В. Інтернет-технології передавання мовних сигналів / Б.В. Дурняк, О.В.
5. Тимченко, Р.С. Колодій, В.І. Сабат. – Львів: Видавництво Української академії друкарства, 2010. – 256 с.
6. Кадемія М. Ю. Соціальні сервіси Веб 2.0 і Веб 3.0 у навчальній діяльності : навчальний посібник / М. Ю. Кадемія, М. М Козяр, В. М. Кобися, М. С. Коваль.–Вінниця: ТОВ «Планер», 2010. – 230 с.
7. Рудий Є.М. Технології передачі дискретних повідомлень. Т.1: Стиснення сигналів: підручник. – Одеса: ОНАЗ ім. О.С. Попова, 2011. – 170 с.
8. Мелешко М.А., Денисенко С.М. Пясківський М.І. Застосування 3D-моделей в мультимедійних електронних освітніх ресурсах. Проблеми інформатизації та управління: Зб. наук. праць. Випуск 3(51). – К.: НАУ, 2015. - с. 86-91.
9. Масловський Б. Г., Дрововозов В. І., Михальчук І. Комп'ютерні побутові мультимедійні системи: навчальний посібник/ МОН України, Національний авіаційний університет. – Київ: НАУ-друк, 2010. – 280 с.
10. А. І. Українець, В. В. Самсонов, Ю. П. Чаплінський, В. В. Кот. Підготовка електронних навчально-методичних ресурсів навчальної дисципліни : Метод. посіб. – Київ : НУХТ, 2009.– 56 с.
11. Іванова О.В. Основи укладання галузевих глосаріїв. Курс лекцій. Навчальний посібник. Третє видання,доповнене. К.: ТОВ «ЦП» Компринт», 2011. – 227 с.
12. Пометун О. І. Інтерактивні технології навчання: Наук.-метод. посібн. / О. І. Пометун, Л. В. Пироженко. — К.: А.С.К., 2004. — 192 с.

13. Баркова О.В. Електронні ресурси як об'єкти універсальної елетронної бібліотеки // Бібліотекознавство. Документознавство. Інформологія. − 2004. − N 2. − с. 75-80.
14. Міжнародний стандарт ISO 14915 «Дизайн (проектування) мультимедіа інтерфейсу користувача – Ергономічні вимоги для інтерактивного мультимедіа інтерфейсу людина–комп'ютер» (BS EN ISO 14915-1:2002.

Інформаційні ресурси в Інтернет

1. http: //irnl/nau.edu.ua – наукова періодика НАУ.
2. http: //er/nau.edu.ua - електронний репозитарій НАУ.
3. Кінаш Р. Система для укладання комп'ютерних версій словників PolyDic ML 3.0: функції та засоби редактора / Роман Кінаш, Роман Мисак, Юрій Каличак, Олександр Мельник // Збірник наукових праць: «Проблеми української термінології». − 2010. − С. 38–42. http://tc.terminology.lp.edu.ua/TK_Zbirnyk_2010/TK_Zbirnyk_2010_kinash_mysak_kalychak_.htm
4. Software ergonomics for multimedia user interfaces. Design principles and framework. – British Standard/European Standard/International Organization for Standardization/10-Dec-2002/24 pages. – ISBN: 0580409236).
5. Margherita Pagani. Encyclopedia Of Multimedia Technology and Networking. I-LAB Centre for Research on the Digital Economy, Bocconi University, Italy. – 1167 p. (електронний варіант є в базі кафедри «Student»).

Електронне навчальне видання

ТЕХНОЛОГІЇ ЕЛЕКТРОННИХ МУЛЬТИМЕДІЙНИХ ВИДАНЬ

Лабораторний практикум
для здобувачів вищої освіти ОС «Бакалавр»
Галузь знань: 18 Виробництво та технології
Спеціальність: 186 Видавництво та поліграфія
Освітньо-професійна програма:
Технології електронних мультимедійних видань

Укладачі:
МЕЛЕШКО Микола Андрійович
ГНІДЕНКО Ірина Андріїна
РАКИЦЬКИЙ Вадим Андрйович

Видавництво НАУ
03058, м. Київ, проспект Гузара Любомира, 1.