МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КІРОВОГРАДСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

ЦЕНТР ЗАОЧНОЇ ТА ДИСТАНЦІЙНОЇ ОСВІТИ

КАФЕДРА ПРОГРАМУВАННЯ ЗА ЗАХИСТУ ІНФОРМАЦІЇ

Комп'ютерна графіка

Методичні вказівки до виконання контрольних робіт

з елементами кредитно - модульної
системи організації навчального процесу

для студентів заочної форми навчання (повні, друга вища)
за напрямками підготовки 6.050102 «Комп'ютерна інженерія»
7.05010202 «Системне програмування»
7.05010201 «Комп'ютерні системи та мережі»

Укладачі:
к.т.н, доцент Н.В. Смірнова
к.т.н, доцент В.В. Смірнов

Кіровоград 2015

Комп'ютерна графіка: Методичні вказівки до виконання контрольних робіт для студентів заочної форми навчання/ Укл.: Смірнова Н.В. Смірнов В.В. - Кіровоград: КНТУ, 2015. – 27 с.

Укладачі:

Смірнова Наталя Володимирівна, к.т.н., доцент кафедри ПЗІ.

Смірнов Володимир Вікторович, к.т.н., доцент кафедри ПЗІ,

Для студентів заочної форми навчання, що вивчають навчальну дисципліну "Комп'ютерна графіка" за напрямками підготовки 6.050102 «Комп'ютерна інженерія», 7.05010202 «Системне програмування», 7.05010201 «Комп’ютерні системи та мережі».

У стислій формі викладені основні принципи побудови об'єктно-орієнтованих подієво-керованих інтерактивних графічних додатків мовою програмування Java у середовищі розробки NetBeansIDE.

Представлені практичні рішення навчальних завдань для кращого освоєння досліджуваного матеріалу, представлені варіанти навчальних завдань для самостійного придбання практичних навичок.

ЗМІСТ

Опис навчальної дисципліни
" Комп'ютерна графіка "

Основна мета курсу полягає в придбанні знань і навичок розробки та побудови об'єктно-орієнтованих подійно-керованих інтерактивних графічних Java додатків із застосуванням сучасних технологій і інструментальних засобів.

У результаті проведення лекцій студенти повинні одержати теоретичні знання і методику ефективної роботи із сучасними методами створення об'єктно-орієнтованих керованих подіями графічних додатків.

Завдання вивчення дисципліни

- Вивчення теоретичних основ проектування додатків комп'ютерної графіки;
- Вивчення теоретичних основ програмування додатків комп'ютерної графіки;
- Вивчення теоретичних основ методів створення графічних 2D - об'єктів;
- Вивчення теоретичних основ методів створення графічних 3D - об'єктів;
- Розв'язок завдань створення елементів керування графічними 2D і 3D об'єктами з використанням платформи JavaFx;
- Придбання практичних навичок в області програмування 2D і 3D додатків комп'ютерної графіки на основі технології Java.

Предметом навчальної дисципліни є створення об'єктно-орієнтованих подійно-керованих інтерактивних додатків комп'ютерної графіки в середовищі програмування NetBeansIDE на платформі Java.

У результаті вивчення навчальної дисципліни студент повинен

знати:

1. Основи створення 2D і 3D графічних об'єктів.
2. Принципи створення подійно-керованих програм.
3. Основи керування 2D і 3D графічними об'єктами.

уміти:

1. Вирішувати завдання створення 2D і 3D додатків комп'ютерної графіки на основі технології Java.
2. Вирішувати завдання створення об'єктно-орієнтованих додатків комп'ютерної графіки в середовищі розробки NetBeansIDE на платформі JavaFx.

3. Вирішувати завдання керування 2D і 3D графічними об'єктами в додатках комп'ютерної графіки.

Контрольні роботи по дисципліні "Комп'ютерна графіка"

№ заняття	Теми контрольних робіт	Кількість годин
	5 семестр	
1.	Перетворення графічних примітивів 2D графіки	2
2.	Побудова і перетворення об'єктів 3D графіки	2
3.	Режими візуалізації об'єктів 3D графіки	2
4.	Анімація об'єктів комп'ютерної графіки. Клас PathTransition і RotateTransition	2
	усього годин	8

3. Шкала оцінювання

За шкалою ECTS	За національною шкалою	За шкалою навчального закладу
A	відмінно	90-100
B-C	добре	75-89
D-E	задовільно	60-74
F-X	незадовільно з можливістю повторної здачі	35-59
F	незадовільно з обов'язковим повторним курсом	1-34

Контрольна робота №1

Тема: Перетворення графічних примітивів 2D графіки

Ціль: Одержати навички роботи з перетвореннями графічних примітивів 2D графіки.

Завдання:

- Створити проект.
- Створити графічний примітив відповідно до варіанта.
- Створити елементи керування для графічного примітива.
- Вибрати колір заливки.
- Виконати програму.

Теоретичні відомості:

Клас `avafx.scene.transform` забезпечує набір зручних класів для виконання обертання, масштабування, зрушення, і перетворення для `Affine` об'єктів.

Графічний примітив має визначені властивості, наприклад:

```
translateXProperty() тип double
translateYProperty() тип double
translateZProperty() тип double
```

Для зв'язку властивості примітива з елементом керування, наприклад, зі слайдером, необхідно використовувати метод `bind()`.

Наприклад, зв'яжемо властивість `scaleXProperty()` примітива `ellipse` с движком `слайдера slider_scale`:

```
//=== прив'язати властивість scale до слайдеру
ellipse.scaleXProperty().bind(slider_scale.valueProperty());
ellipse.scaleYProperty().bind(slider_scale.valueProperty());
```

Для спрощення побудови сцени слід використовувати контейнери для об'єктів сцени, такі, як вертикальний і горизонтальні бокси. Створити бокси можна в такий спосіб:

```
//=== бокс для слайдерів
VBox control_vbox = new VBox();    //контейнер для об'єкта
control_vbox.setLayoutX(100);      //відображати в координатах
control_vbox.setLayoutY(100);
control_vbox.setSpacing(10);


//=== бокс для міток
VBox label_vbox = new VBox();      //контейнер для об'єкта
label_vbox.setLayoutX(20);         //відображати в координатах
label_vbox.setLayoutY(100);
label_vbox.setSpacing(30);

//====== помістити мітки у контейнер
label_vbox.getChildren().addAll(lbl_opacity,lbl_scale,lbl_rotate,
lbl_translateX,lbl_translateY);

//====== помістити слайдериу контейнер
control_vbox.getChildren().addAll(slider_opacity,slider_scale,
slider_rotate,slider_translateX,slider_translateY);

//====== помістити всі вузли у корінь
root.getChildren().addAll(label_vbox,control_vbox);

//====== помістити еліпс у корінь
root.getChildren().add(ellipse);
```

У методі `start()` створити сцену і включити в неї групу `root`:

```
//====== створити головну сцену
Scene main_scene = new Scene(root, 1000, 500,Color.TRANSPARENT);
```

Методичні вказівки до виконання контрольної роботи:

– Використовуючи результати лабораторних робіт №1 і №2 створити головне вікно, об'єкти керування, графічний примітив і розмістити об'єкти в сцені.

Елементи керування повинні здійснювати наступні перетворення:

– Прозорість – `Opacity`.

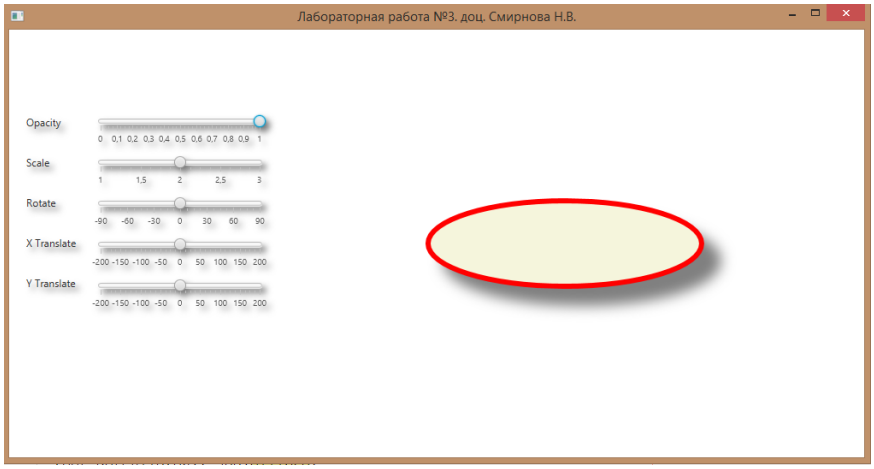
– Масштабування – `Scale`.

– Поворот – `Rotate`.

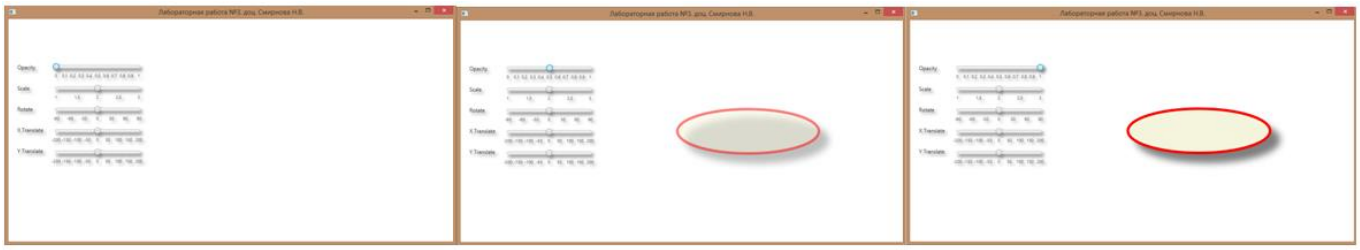
– Переміщення по осях `X` і `Y` – `Translatex,Y`.

– Виконати програму.

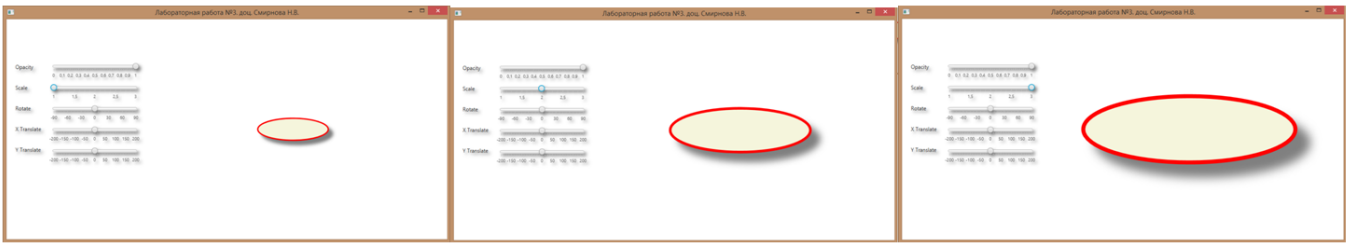
Результат виконання програми:



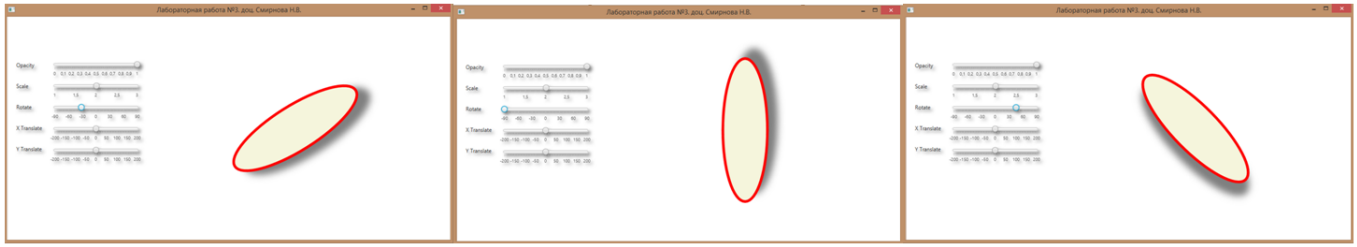
Реалізація властивості **Opacity**



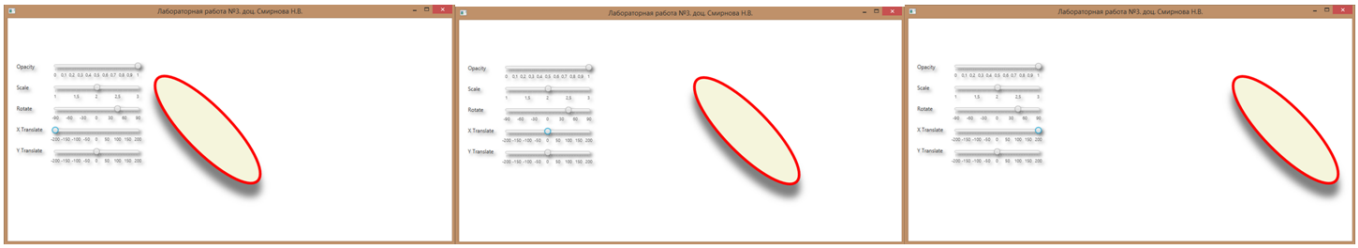
Реалізація перетворення **Scale**



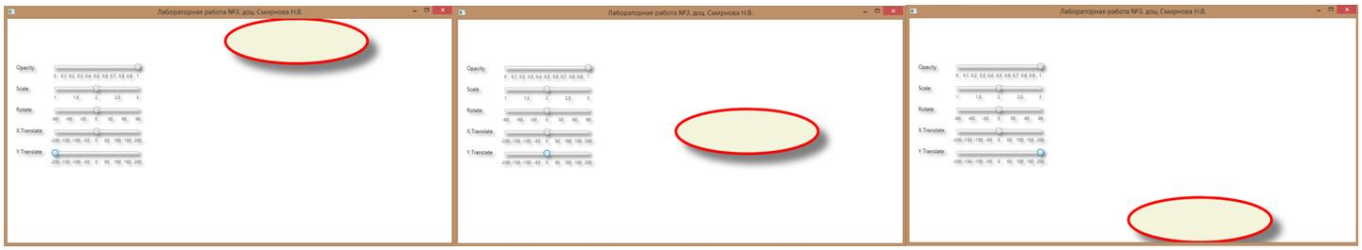
Реалізація перетворення **Rotate**



Реалізація перетворення **Translate X**



Реалізація перетворення **Translate Y**



Контрольні питання:

1. Призначення боксів – контейнерів.
2. Порядок створення слайдера.
3. Властивості примітива `property()`.
4. Як здійснюється зв'язування властивостей графічного примітива та елемента керування.

Зміст звіту:

1. Лістинг програми.
2. Висновки за результатами роботи.

Варіанти:

	еліпс	**коло**	**прямокутник**
0	+		
1		+	
2			+
3	+		
4		+	
5			+
6	+		
7		+	
8			+
9	+		

В заголовку вікна вказати № контрольної роботи, групу, ПІБ студента

Примітки: файл програми, що виконується, з розширенням *.jar і демонстраційний ролік з розширенням *.wmv знаходяться у папці “Приклади виконання”.

Контрольна робота № 2

Тема: Побудова і перетворення об'єктів 3D графіки

Одержати навички роботи зі створенням і перетвореннями об'єктів 3D графіки.

Завдання:

- Створити проект.
- Створити об'єкт 3D графіки відповідно до варіанта.
- Створити елементи керування для створеного об'єкта.
- За допомогою елементів керування вибрати осі повороту об'єкта.
- За допомогою слайдера здійснювати поворот об'єкта по осях X,Y,Z.
- Виконати програму.

Теоретичні відомості:

Конструктор класу `Box` має такий вигляд:

```
Box(double width, double height, double depth);
```

Методи класу `Box`:

Тип	Метод
DoubleProperty	depthProperty()
depthproperty	getDepth()
double	getHeight()
double	getWidth()
DoubleProperty	heightProperty()
void	setDepth(double value)
void	setHeight(double value)
void	setWidth(double value)
DoubleProperty	widthProperty()

Конструктор класу `Cylinder` має такий вигляд:

```
Cylinder(double radius, double height)
```

Методи класу `Cylinder`:

Тип	Метод
int	getDivisions()
double	getHeight()
double	getRadius()
DoubleProperty	heightProperty()
DoubleProperty	radiusProperty()
void	setHeight(double value)
void	setRadius(double value)

Конструктор класу `Sphere` має такий вигляд:

```
Sphere(double radius)
```

Методи класу `Sphere`:

Тип	Метод
int	getDivisions()
double	getRadius()
DoubleProperty	radiusProperty()
void	setRadius(double value)

Вісь обертання задається об'єктом `Point3D`:

```
//====== вісь обертання об'єкта
Point3D p3d_cylinder1;
Point3D p3d_cylinder2;
Point3D p3d_box;
```

Тінь для елементів керування створюється методом:

```
//====== створення тіні
DropShadow dropShadow = new DropShadow(10, 5, 5, Color.GRAY);
```

Ефект тіні може застосовуватися як до окремих елементів керування,

```
Label lbl_1 = new Label("Узел 1");
lbl_1.setEffect(dropShadow);
```

так і до контейнерів:

```
//=== бокс для елементів керування label
HBox label_hbox = new HBox();
label_hbox.setLayoutX(160);        //відображати в координатах
label_hbox.setLayoutY(330);
label_hbox.setSpacing(380);
//
label_hbox.setEffect(dropShadow);
```

В останньому випадку, до всіх елементів контейнера буде застосований ефект.

Приклад створення елементів управління класу `CheckBox`:

```
cb_cylinder_X = new CheckBox("X");
cb_cylinder_Y = new CheckBox("Y");
cb_cylinder_Z = new CheckBox("Z");
```

Методичні вказівки до виконання контрольної роботи:

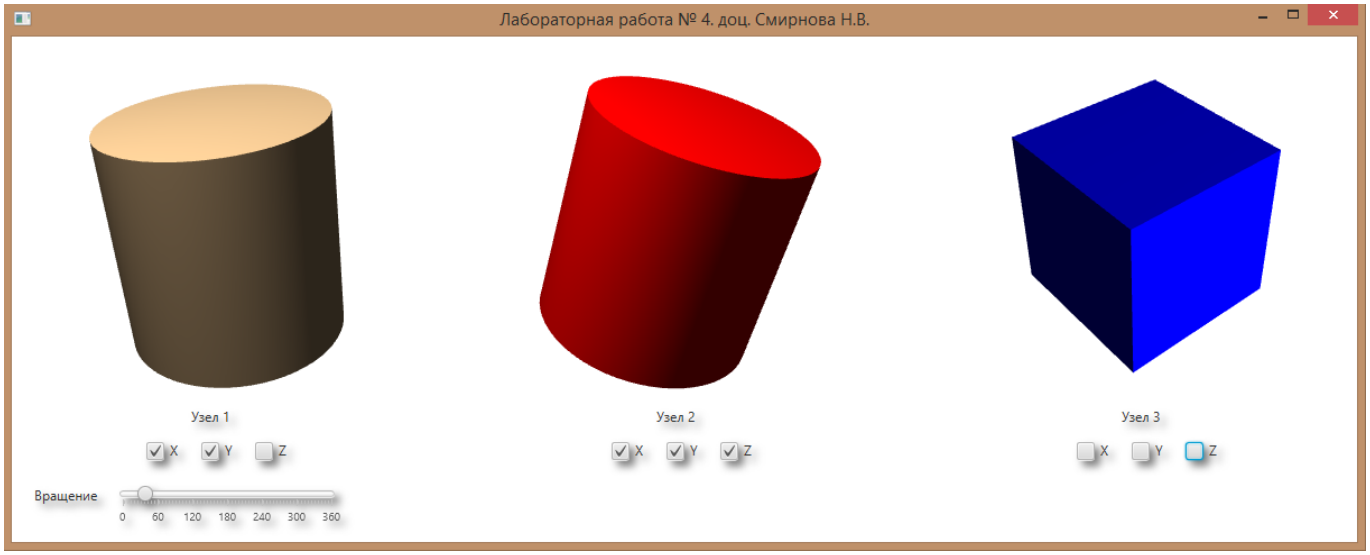
Використовуючи результати попередніх контрольних і лабораторних робіт створити головне вікно, об'єкти керування, об'єкт 3D графіки і розмістити об'єкти в сцені.

При цьому використовувати клас `Sub` пакета `swcFX_lib`, який необхідно скопіювати в папку `src` проекту. Це дасть можливість у головній сцені розміщати безліч підсцен, які не будуть впливати одна на іншу.

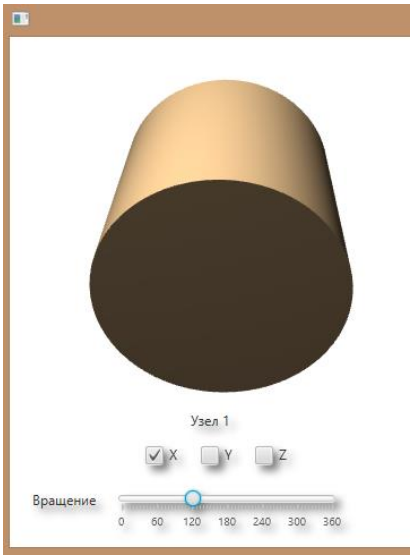
У кожної підсцені встановлюється своя камера, джерела висвітлення і осі трансформації об'єкта.

- Елементи керування повинні здійснювати наступні перетворення:
- Поворот 3D об'єкта по осі X.
- Поворот 3D об'єкта по осі Y.
- Поворот 3D об'єкта по осі Z.
- Поворот 3D об'єкта по осі X,Y,Z.

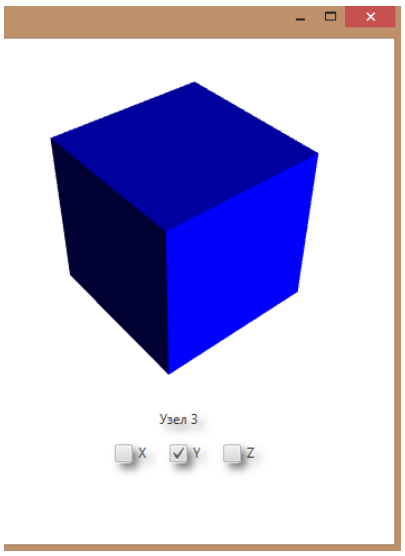
Результат виконання програми:



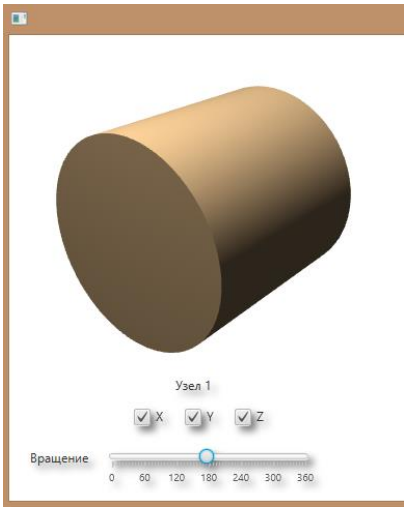
Поворот 3D об'єкта по осі X



Поворот 3D об'єкта по осі Y



Поворот 3D об'єкта по осі X,Y,Z.



Контрольні питання:

1. Порядок створення 3D об'єктів.
2. Властивості 3D об'єктів.
3. Призначення об'єкта `Subscene`.
4. Порядок призначення осей трансформації 3D об'єкту.
5. Як визначається матеріал заливки 3D об'єктів.

Зміст звіту:

У звіті повинні бути представлені:

1. Лістинг програми.
2. Висновки за результатами роботи.

Варіанти:

	Сфера	Циліндр	Куб
0	+		
1		+	
2			+
3	+		
4		+	
5			+
6	+		
7		+	
8			+
9	+		

В заголовку вікна вказати № контрольної роботи, групу, ПІБ студента

Примітки: файл програми, що виконується з розширенням *.jar і демонстраційний ролік з розширенням *.wmv знаходяться у папці “Приклади виконання”.

Контрольна робота № 3

Тема: Режими візуалізації об'єктів 3D графіки

Ціль: Одержати навички роботи зі створенням і перетвореннями об'єктів 3D графіки.

Завдання:

– Створити проект.

– Створити об'єкт 3D графіки відповідно до варіанта.

– Створити елементи керування для створеного об'єкта.

– За допомогою елементів керування вибрати режими візуалізації об'єкта.

– Підготувати файл *.jpg у якості текстури.

– Здійснити візуалізацію об'єкта в режимах .LINE, .FILL і текстури.

– За допомогою слайдера здійснювати поворот об'єкта по осях X,Y,Z.

Теоретичні відомості:

– Використовуючи результати попередніх контрольних і лабораторних робіт створити головне вікно, об'єкти керування, об'єкт 3D графіки і розмістити об'єкти в сцені.

При цьому використовувати клас `Sub` пакета `swcFX_lib`, який необхідно скопіювати в папку src проекту

Створення матеріалу заливання для об'єктів 3D здійснюється в такий спосіб:

```
        //================================================================
        // Створити матеріал заливання
        //================================================================
        private void createFillMaterial() {
           //=== сфера
           sphere_material_fill = new
    PhongMaterial(sphere_ColorPicker.getValue());
           //=== бокс
           box_material_fill = new
PhongMaterial(box_ColorPicker.getValue());
           //=== для режима LINE
           black_material_line = new PhongMaterial(Color.BLACK);
        }
```

Створення об'єктів класу ColorPicker здійснюється в такий спосіб:

```
//=============================================================
// Створити ColorPicker
//=============================================================
private void createColorPicker() {
    //=== цилиндр
    cylinder_colorPicker = new ColorPicker(Color.GRAY);
    cylinder_colorPicker.setMaxWidth(0);
    //=== сфера
    sphere_colorPicker = new ColorPicker(Color.RED);
    sphere_colorPicker.setMaxWidth(0);
    //=== бокс
    box_colorPicker = new ColorPicker(Color.BLUE);
    box_colorPicker.setMaxWidth(0);
}
```

Обробка події елемента керування `ColorPicker`, установка кольору матеріалу і режиму візуалізації здійснюється в оброблювачі події `setOnAction()`:

```
//=== cylinder_colorPicker
cylinder_colorPicker.setOnAction((ActionEvent t) -> {
     //=== одержати колір
     cylinder_material_fill = new
PhongMaterial(cylinder_colorPicker.getValue());
     cylinder.setMaterial(cylinder_material_fill); //установити матеріал
     cylinder.setDrawMode(DrawMode.FILL);          //режим заливання
     resetRadio();                                 //скинути радіобокс
});
```

Завантаження матеріалу текстури для об'єктів 3D здійснюється з файлу з розширенням *.jpg в об'єкт класу `Image` за допомогою ініціалізації його конструктора:

```
box_image = new Image(Lab_5_Smirnova_N_V.class.getResource("KNTU.jpg")
      .toExternalForm());                    //завантажити рисунок з
ресурсу
      box_material_texture = new PhongMaterial();    //створити матеріал
      box_material_texture.setDiffuseMap(box_image);//призначити йому об'єкт
Image
```

Створення перемикачів режиму здійснюється в такий спосіб:

```
//напис
Label lbl_radio = new Label("В Візуалізація:");
//
ToggleGroup tg = new ToggleGroup();
rb_fill = new RadioButton("Fill");
rb_fill.setToggleGroup(tg);
//
rb_line = new RadioButton("Line");
```

```
        rb_line.setToggleGroup(tg);
        rb_line.setSelected(true);
        //
        rb_texture = new RadioButton("Текстура");
        rb_texture.setToggleGroup(tg);
```

Перемикання режимів візуалізації здійснюється за допомогою методу `setViewMode()`:

```
        //=============================================================
        // Установити режим відображення LINE/FILL
        //=============================================================
        private void setViewMode(int mode) {
            //=== режим FILL
            if (mode == MODE_FILL) {
                cylinder.setDrawMode(DrawMode.FILL);
                sphere.setDrawMode(DrawMode.FILL);
                box.setDrawMode(DrawMode.FILL);
            } else {
                cylinder.setDrawMode(DrawMode.LINE);
                sphere.setDrawMode(DrawMode.LINE);
                box.setDrawMode(DrawMode.LINE);
            }
        }
```

Методичні вказівки до виконання контрольної роботи:

Елементи керування повинні здійснювати наступні перетворення з попередньої контрольної роботи:

- Поворот 3D об'єкта по осі X.
- Поворот 3D об'єкта по осі Y.
- Поворот 3D об'єкта по осі Z.
- Поворот 3D об'єкта по осі X,Y,Z.

І реалізовувати додаткові функції:

- Вибір кольору заливання і текстури
- Режим відображення об'єкта.

Для реалізації завдання:

- Створити проект.
- Створити сцену.
- Створити елементи керування.
- Створити графічний 3D об'єкт відповідно до варіанта.

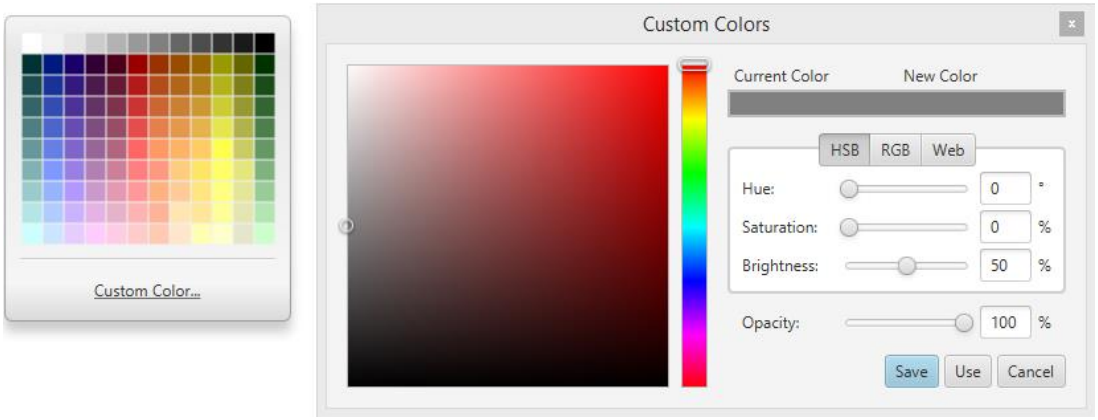
- Помістити об'єкт у субсцену.
- Субсцену помістити в головну сцену.
- Запустити і налагодити проект.

Приклад оформлення методу start():

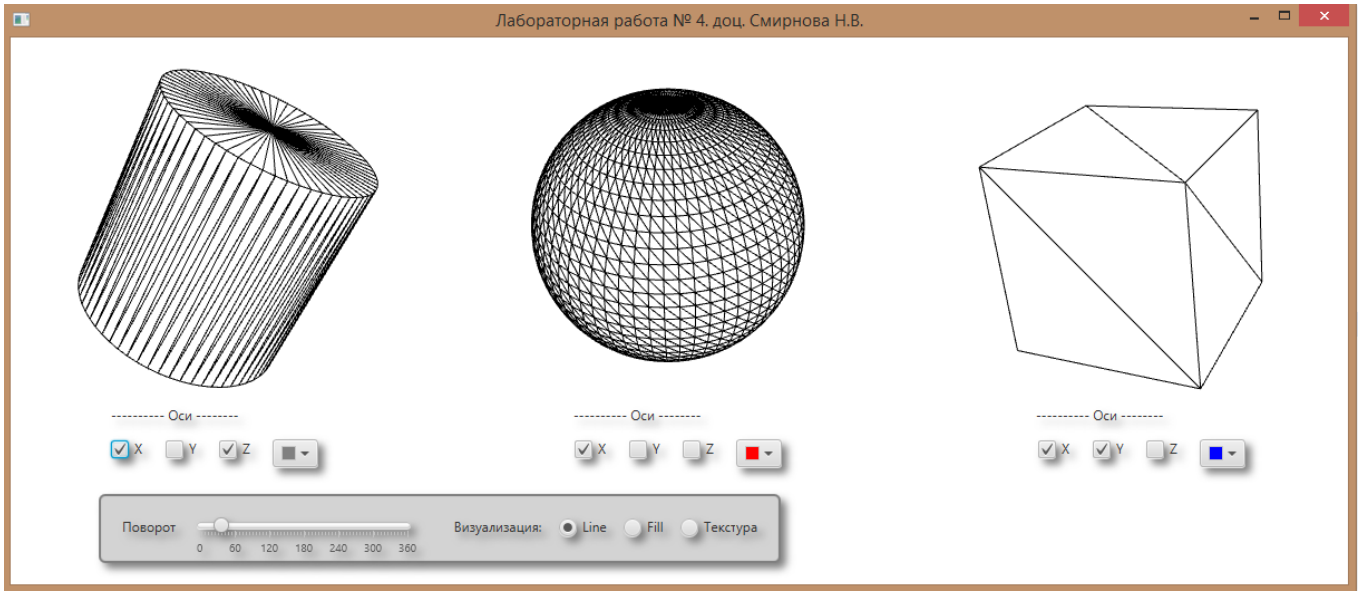
```
//================================================================
// Початок
//================================================================
@Override
public void start(Stage stage) {
   //====== заголовок вікна
   stage.setTitle("Контрольна робота № 5. ст. Петров І.С.");
   //====== створити головну сцену
   Scene main_scene = new Scene(root, 1200, 490, Color.TRANSPARENT);
   //====== створити вузли сцени с елементами керування
   CreateControlNodes();
   //====== створити вузли сцени с графікою
   CreateGraphNodes();
   //====== помістити головну сцену у вікно
   stage.setScene(main_scene);
   //====== відобразити вікно
   stage.show();

}
```

Результат виконання програми:

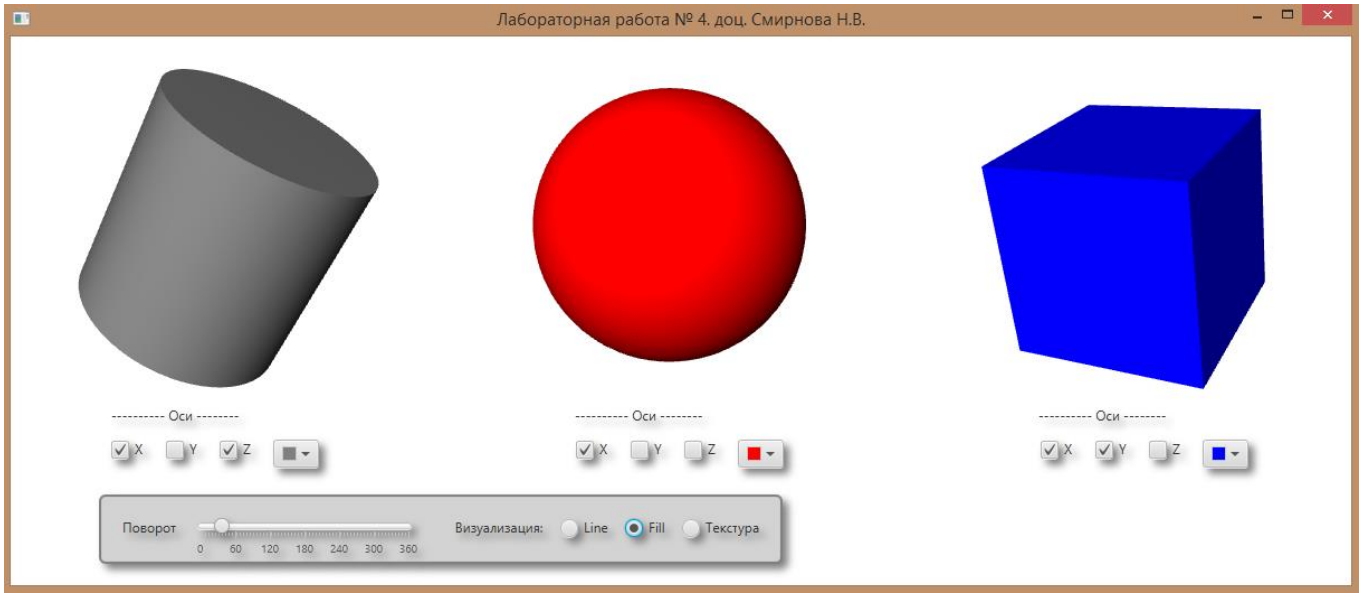
Вибір кольору заливання за допомогою елемента ColorPicker:



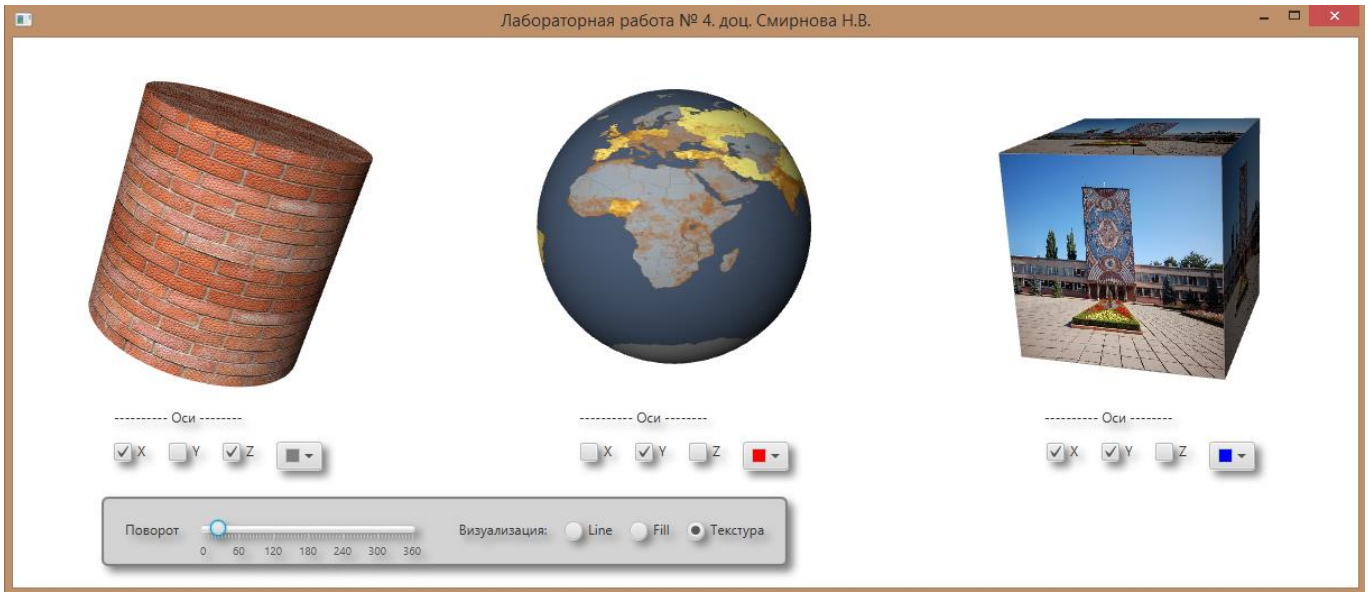
Візуалізація об'єктів у режимі .LINE



Візуалізація об'єктів у режимі .FILL:



Візуалізація об'єктів у режимі заливання текстурою:



Контрольні питання:

1. Призначення і створення елемента керування `ColorPicker`.
2. Порядок створення і завантаження текстур для 3D об'єкта.
3. Порядок оголошення матеріалу для заливання.
4. Керування режимами візуалізації `.LINE`, `.FILL` і накладення текстури.
5. Порядок створення субсцени.

Зміст звіту:

У звіті повинні бути представлені:

- Структурна схема програми
- Лістинг програми.
- Висновки за результатами роботи.

Варіанти:

	Сфера	**Циліндр**	**Куб**
0	+		
1		+	
2			+
3	+		
4		+	
5			+
6	+		
7		+	
8			+
9	+		

В заголовку вікна вказати № контрольної роботи, групу, ПІБ студента

Примітки: файл програми, що виконується, з розширенням *.jar і демонстраційний ролік з розширенням *.wmv знаходяться у папці “Приклади виконання”.

Контрольна робота № 4

Тема: Анімація об'єктів комп'ютерної графіки. Клас PathTransition і RotateTransition

Ціль: Одержати навички роботи з анімацією об'єктів комп'ютерної графіки.

Завдання:

- Створити проект.
- Створити об'єкт комп'ютерної графіки відповідно до варіанта.
- Створити екземпляр класів анімації об'єкта.
- Запустити анімацію.

Теоретичні відомості:

Платформа JavaFx забезпечує створення двох видів анімації – анімацію по ключових кадрах і анімацію із вбудованою тимчасовою шкалою.

JavaFx-анімацію представляє пакет `javafx.animation`, базовим класом якого є клас `Animation`. Клас `Animation` розширюється класами `Timeline` і `Transition`, при цьому клас `Timeline` представляє *анімацію по ключових кадрах*, а клас `Transition` – *анімацію з вбудованою тимчасовою шкалою*.

Клас `Animation` має набір властивостей, що дозволяють управляти швидкістю і напрямком анімації, затримкою і кількістю циклів анімації, встановлювати автореверс анімації, зчитувати статус анімації, обробляти завершення анімації та ін.

Швидкість і напрямок анімації можна встановити за допомогою методу `setRate(double value)`, затримку анімації – за допомогою методу `setDelay (Duration value)`, кількість циклів анімації – методом `setCycleCount (int value)`, автореверс анімації – методом `setAutoReverse (boolean value)`, вважати статус статус `Animation.Status.PAUSED`, `Animation.Status.RUNNING` або `Animation.Status.STOPPED` анімації – методом `getStatus()`, установити оброблювач завершення анімації – методом `setOnFinished(EventHandler<ActionEvent>value)`.

Також клас `Animation` надає методи керування життєвим циклом анімації:

- `jumpTo(Duration time)` – перехід анімації до зазначеної позиції на тимчасовій шкалі;
- `playFrom(Duration time)` – запуск анімації, починаючи із зазначеної позиції на тимчасовій шкалі;

- `play()` – запуск анімації з поточної позиції на тимчасовій шкалі;
- `playFromStart()` – запуск анімації з первісної позиції на тимчасовій шкалі;
- `stop()` – зупинка анімації;
- `pause()` – пауза анімації.

Спосіб зміни значення JavaFx-властивості протягом анімації представлений класом `javafx.animation.Interpolator`, що мають статичні поля:

- `Interpolator.DISCRETE` – дискретна зміна значення JavaFx-властивості, при якому значення залишається початковим до закінчення тимчасового інтервалу, коли значення стає кінцевим;
- `Interpolator.LINEAR` (за замовчуванням) – лінійна зміна значення JavaFx-властивості, при якому значення визначається по формулі:

  `startValue + (endValue - startValue) × fraction`;
- `Interpolator.EASE_BOTH` – використовується величина `0.2` для приросту і зменшення значення JavaFx-властивості;
- `Interpolator.EASE_IN` – використовується величина `0.2` для приросту значення

  JavaFx-властивості;
- `Interpolator.EASE_OUT` – використовується величина `0.2` для зменшення значення

  JavaFx-властивості.

Крім того, можна створити користувацький клас `Interpolator` з перевизначенням його методів, що описують зміну значення JavaFx-властивості.

`Transition`- анімація з вбудованою тимчасовою шкалою також використовує об'єкт `Interpolator` у якості значення властивості `interpolator` класу `Transition`.

Методичні вказівки до виконання контрольної роботи:

Після створення проекту, у проекті необхідно реалізувати наступне:

- Створити геометричну фігуру як шлях для анімації. Цим шляхом буде переміщатися об'єкт анімації.
- Створити об'єкт класу `Path` і в нього помістити геометричну фігуру.
- Створити об'єкт класу `PathTransition` і в нього помістити об'єкт класу `Path`.
- Запустити анімацію

Приклад створення дуги:

```
MoveTo move_to = new MoveTo(begin_x, begin_y);
 //====== велика дуга для еліпса
 ArcTo arcTo = new ArcTo();              //створити
 arcTo.setX(begin_x+1);                  //координати по X
(початок)
 arcTo.setY(begin_y);                    //координати по Y
(початок)
 arcTo.setRadiusX(rad_x);                //радіус по X
 arcTo.setRadiusY(rad_y);                //радіус по Y
 arcTo.setLargeArcFlag(true);            //прапор велика дуга
```

Приклад створення екземпляра класу `Path`:

```
 Path path = new Path();                 //створити
 path.setStroke(Color.GRAY);             //колір шляху
 path.setFill(Color.TRANSPARENT);        //не заливати
 path.getStrokeDashArray().setAll(10d, 15d);//пунктир
 //====== додати в шлях траєкторію
 path.getElements().add(move_to, arcTo);
```

Приклад створення екземпляра класу `PathTtransition`:

```
path_transition_1 = new PathTransition(seconds(8), path_1,
circle_1);
//path_transition_1.setOrientation(PathTransition.OrientationTyp
e.
ORTHOGONAL_TO_TANGENT);
 path_transition_1.setCycleCount(Timeline.INDEFINITE);
 path_transition_1.setAutoReverse(false);
```

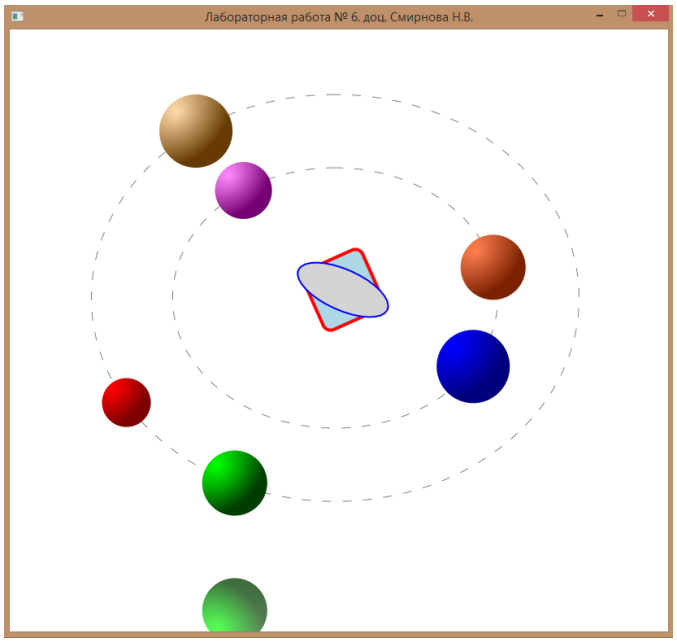
Запуск анімації:

```
public void play() {
    path_transition_1.play();
}
```

Зупинка анімації:

```
@Override
public void stop() {
    path_transition_1.stop();
}
```

Результат виконання програми:



Контрольні питання:

1. Призначення і створення класу `Path`.
2. Призначення і створення класу `PathTransition`.
3. Константи класу Interpolator
4. Порядок створення анімації графічного об'єкта.
5. Керування режимами візуалізації анімації.

Зміст звіту:

У звіті повинні бути представлені:

- Лістинг програми.
- Висновки за результатами роботи.

Варіанти:

	Еліпс	**Сфера**	**Куб**
0	+		
1		+	
2			+
3	+		
4		+	
5			+
6	+		
7		+	
8			+
9	+		

В заголовку вікна вказати № контрольної роботи, групу, ПІБ студента

Примітки: файл програми, що виконується, з розширенням *.jar і демонстраційний ролік з розширенням *.wmv знаходяться у папці “Приклади виконання”.

Рекомендована література

1. http://www.oracle.com/
2. http://docs.oracle.com/apps/search/search.jsp?category=java&product=&q=javafx
3. Johan Vos Pro JavaFX 8. A Definitive Guide to Building. Desktop, Mobile, and Embedded Java Clients / Johan Vos, Weiqi Gao, 2014. – APRESS, 579 с.
4. Carl Dea JavaFX 8 Introduction by Example Second Edition / Carl Dea Mark Heckler, 2014. – APRESS, 383 с.
5. Тимур Машнин JavaFX 2.0: разработка RIA-приложений / Тимур Машнин – СПб.: БХВ-Петербург, 2012. - 320 с.

Методичне забезпечення

1. Приходькіна А.І., Смірнова Н.В. Робоча програма навчальної дисципліни для студентів заочної форми навчання / Укл.: Приходькіна А.І., Смірнова Н.В. – Кіровоград: КНТУ, 2015.
2. Приходькіна А.І. Комп'ютерна графіка: Лекції для студентів заочної форми навчання / Укл.: Приходькіна А.І. – Кіровоград: КНТУ
3. Смірнова Н.В. Комп'ютерна графіка: Методичні вказівки до виконання лабораторних робіт для студентів заочної форми навчання / Укл.: Смірнова Н.В. Смірнов В.В. –Кіровоград: КНТУ, 2015.
4. Смірнова Н.В. Комп'ютерна графіка: Методичні вказівки до виконання контрольних робіт для студентів заочної форми навчання / Укл.: Смірнова Н.В. Смірнов В.В. – Кіровоград: КНТУ, 2015.
5. Смірнова Н.В. Комп’ютерна графіка: методичні вказівки до виконання курсового проектування для студентів заочної форми навчання / Укл.: Смірнова Н.В., Смірнов В.В. – Кіровоград: КНТУ, 2015.
6. Смірнова Н.В. Комп’ютерна графіка: Методичні вказівки до виконання самостійних робіт для студентів заочної навчання/ Укл.: Смірнова Н.В., Смірнов В.В. – Кіровоград: КНТУ, 2015.
7. Смірнов.В.В., Смирнова Н.В. Навчальні відеоролики з прикладами реалізації програм 2D і 3D комп'ютерної графіки http://forum.kntu.kr.ua/viewtopic.php?f=633&t=8051&sid=40e4eb7fca9dc189965c7bf352a3a36d

