ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ ДЛЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Лабораторная работа № 8. Разработка приложений для обработки графики, анимации и жестов на языке Swift

Table of Contents

ВВЕДЕНИЕ

Данные методические указания предназначены для выполнения лабораторных работ по дисциплине «Технологии программирования для мобильных приложений» студентами 2 курса специальности «Прикладная информатика».

Структура лабораторной работы

Каждая лабораторная работа содержит тексты задач и контрольные вопросы, ответы на которые проверяются преподавателем при приеме работы у студента. Для некоторых задач приводятся указания по их решению.

Каждая задача имеет уникальный в пределах данной лабораторной работы номер.

В приложение вынесены требования и рекомендации по оформлению исходных текстов программ.

Порядок сдачи лабораторной работы

Выполнение студентом лабораторной работы и сдача ее результатов преподавателю происходит следующим образом.

1. Студент выполняет разработку программ.

В ходе разработки студент обязан следовать указаниям к данной задаче (в случае их наличия). Исходные тексты программ следует разрабатывать в соответствии с требованиями к оформлению, приведенными в приложении.

2. Студент выполняет самостоятельную проверку исходного текста каждой разработанной программы и правильности ее работы, а также свои знания по теме лабораторной работы.

Исходные тексты программ должны соответствовать требованиям к оформлению, приведенным в приложении. Недопустимо отсутствие в тексте программы следующих важных элементов оформления: спецификации программного файла и подпрограмм, а также отступов, показывающих структурную вложенность языковых конструкций.

Для проверки правильности работы программы студенту необходимо разработать набор тестов и на их основе провести тестирование программы. Тестовый набор должен включать примеры входных и выходных данных, приведенные в тексте задачи, а также тесты, разработанные студентом самостоятельно.

Самостоятельная проверка знаний по теме лабораторной работы выполняется с помощью контрольных вопросов и заданий, приведенных в конце текста лабораторной работы.

3. Студент защищает разработанные программы. Защита заключается в том, что студент должен ответить на вопросы преподавателя, касающиеся разработанной программы, и контрольные вопросы.

К защите необходимо представить исходные тексты программ, оформленных в соответствии с требованиями, и протоколы тестирования каждой программы, подтверждающие правильность ее работы.

Протокол тестирования включает в себя тест (описание входных данных и соответствующих им выходных данных), описание выходных данных, полученных при запуске программы на данном тесте, и отметку о прохождении теста. Тест считается пройденным, если действительные результаты работы программы совпали с ожидаемыми.

ТРЕБОВАНИЯ К ОТЧЁТУ И ОФОРМЛЕНИЮ КОДА

В файле `Readme` проекта на `github` должна быть ссылка на отчёт. Отчет опубликовать в репозитории в папке `docs` или во внешнем хранилище, добавив ссылку на него в файл `Readme` репозитория.

Отчет по лабораторной работе содержит тексты задач, фрагменты кода, описывающие основную функциональность и скриншот разработанного приложения в симуляторе.

Исходный код приложений на языке Objective-C должен соответствовать модели КИС для Objective-C.

Код проекта должен содержать комментарии в стиле Markdown, комментарии PRAGMA, справочные комментарии (см. лабораторную работу 7) и сведения об авторе.

Исходный код приложений должен быть оформлен согласно руководству стиля в зависимости от языка:

1. Swift:
   a) http://ilya2606.ru/?p=1846
   b) https://github.com/RedMadRobot/RMR-swift-style-guide
   c) https://github.com/raywenderlich/swift-style-guide
2. Objective-C:
   a) https://github.com/DigDes/objective-c-style-guide
   b) https://github.com/raywenderlich/objective-c-style-guide

Репозиторий должен содержать следующие ветки:

1. **main** — файл Readme со ссылкой на отчет и описание веток и выполненных заданий.
2. **example-task1** — каталог проекта example1, содержащий проект для примера 1.1
3. **example-task2** — каталог проекта example2, содержащий проект для примера 1.2
4. **example-task3** — каталог проекта example3, содержащий проект для примера 1.3
5. **example-task4** — каталог проекта example4, содержащий проект для примера 1.4
6. **feature-task2-1** — каталог проекта task2-1, содержащий проект для задачи 2.1.
7. **feature-task2-2** — каталог проекта task2-2, содержащий проект для задачи 2.2.
8. **feature-task2-3** — каталог проекта task2-3, содержащий проект для задачи 2.3.
9. **feature-task2-4** — каталог проекта task2-4, содержащий проект для задачи 2.4.
10. **feature-task3** — каталог проекта task3, содержащий проект для задачи 3.

Структура файла Readme

Файл Readme в каждой ветке должен соответствовать структуре

- Project Name
- Descrition
- Installation

- Usage
- Authors
- Links and Additional Notes

Подробнее о назначении каждого раздела можно узнать на https://www.makeareadme.com.

КРИТЕРИИ ОЦЕНИВАНИЯ

Оценка 4

Выполнены **все примеры для изучения**. Отчёт содержит описание задачи, ключевые фрагменты кода и скриншот приложения в симуляторе или на телефоне. Отчёт должен быть опубликован в репозитории или должна быть ссылка на него в файле `Readme`. Отчёт и исходный код проекта может содержать ошибки. Максимальная задержка на предоставление отчёта, исходного кода проекта, включая публикацию в репозитории и задании курса, а также его защиту — 2 недели.

Оценка 5

Выполнены **все примеры для изучения** и даны **ответы на контрольные вопросы**. Отчёт содержит описание задачи, ключевые фрагменты кода, скриншот приложения в симуляторе или на телефоне и ответы на контрольные вопросы. Отчёт должен быть опубликован в репозитории или должна быть ссылка на него в файле `Readme`. Отчёт и исходный код проекта может содержать ошибки. Максимальная задержка на предоставление отчёта, исходного кода проекта, включая публикацию в репозитории и задании курса, а также его защиту — 2 недели.

Оценка 6-7

Выполнены **примеры для изучения** и реализованы **приложения согласно заданиям 2.1-2.4** и даны **ответы на контрольные вопросы**. Отчёт содержит описание задачи, ключевые фрагменты кода, скриншот приложения в симуляторе или на телефоне и ответы на контрольные вопросы. Отчёт должен быть опубликован в репозитории или должна быть ссылка на него в файле `Readme`. Отчёт и исходный код проекта может содержать ошибки. Максимальная задержка на предоставление отчёта, исходного кода проекта, включая публикацию в репозитории и задании курса, а также его защиту — 1 неделя.

Оценка 8

Рассмотрены **примеры для изучения**, реализованы **приложения согласно заданиям 2.1-2.4, 3.1** и даны **ответы на контрольные вопросы**. Отчёт содержит описание задачи, ключевые фрагменты кода, скриншот приложения в симуляторе или на телефоне и ответы на контрольные вопросы. Отчёт должен быть опубликован в репозитории или должна быть ссылка на него в файле `Readme`. Отчёт и исходный код могут содержать незначительные ошибки. Максимальная задержка на предоставление отчёта, исходного кода проекта, включая публикацию в репозитории и задании курса, а также его защиту — 1 неделя.

Оценка 9

Рассмотрены **примеры для изучения**, реализованы **приложения согласно заданиям 2.1-2.4, 3.1** и даны **ответы на контрольные вопросы**. Отчёт содержит описание задачи, ключевые фрагменты кода, скриншот приложения в симуляторе или на телефоне и ответы на контрольные вопросы. Отчёт должен быть опубликован в репозитории или должна быть ссылка на него в файле `Readme`. Лабораторная работа опубликована в репозитории и в задании курса в срок, а так же защищена в срок. Не содержит ошибок.

1. ПРИМЕРЫ ДЛЯ ИЗУЧЕНИЯ

1.1 Рисование (приложение на языке objective-C)

ЗАДАЧИ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Разобрать пример рисования на языке Objective-C

Входные данные

Цвет, размер и форма кисти, использующейся для рисования.

Выходные данные

Рисунок, состоящий из линий произвольной формы.

Указания

Создайте новый Single View Application проект. Добавьте в интерфейс приложения компонент UIImageView (Рис. 1)

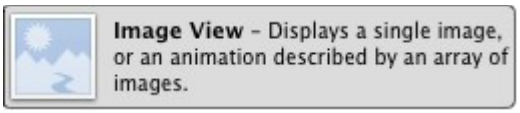

Рис. 1. Компонент холста

Создайте IBOutlet области рисования в контроллере. Удерживая нажатой клавишу Ctrl перетащите UIImageView в область interface файла ViewController.m используя режим Assistant editor (Рис. 2).

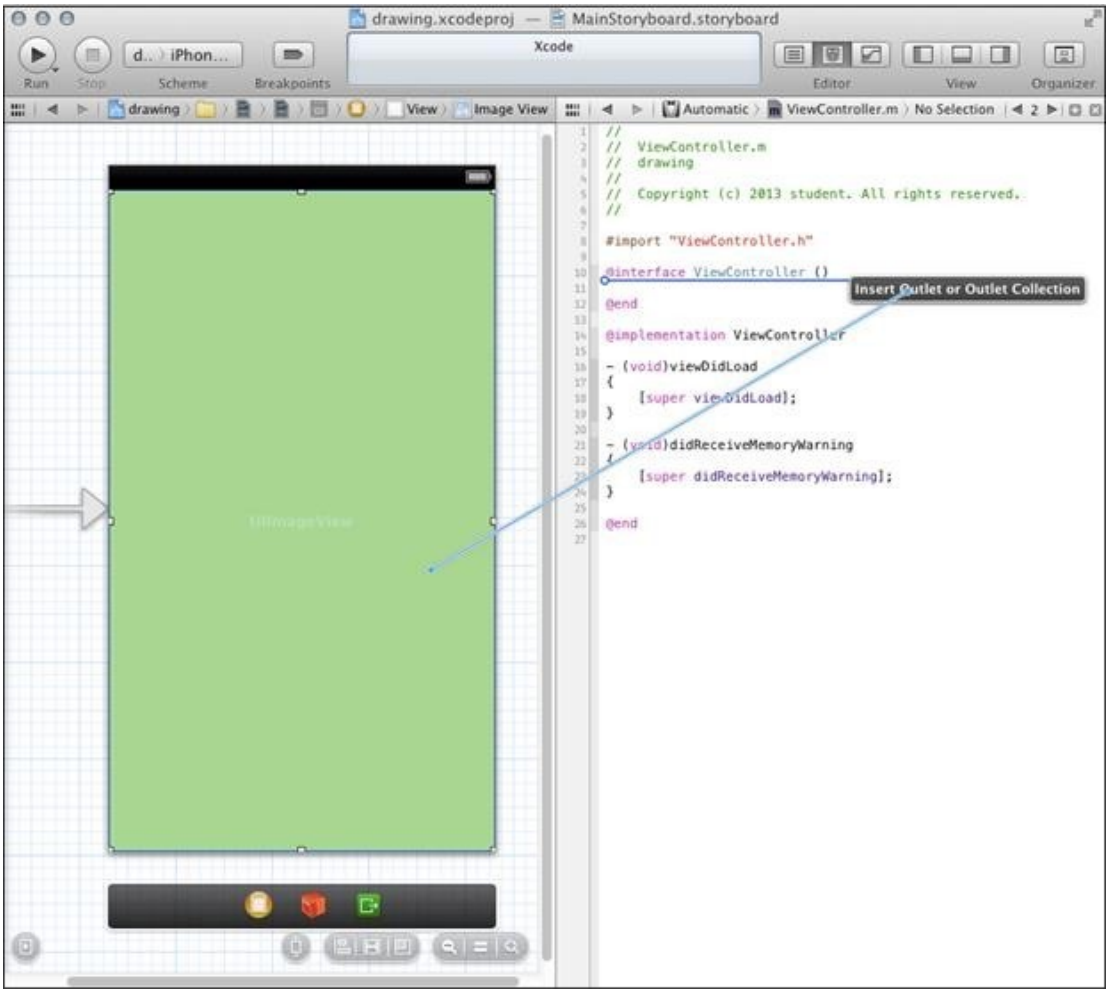

Рис. 2. Добавление связи

Введите canvas и нажмите Connect (Рис. 3).
В область interface добавиться объявление холста (Листинг 1).

```
@property (strong, nonatomic) IBOutlet UIImageView *canvas;
```

Листинг 1. Объявление холста

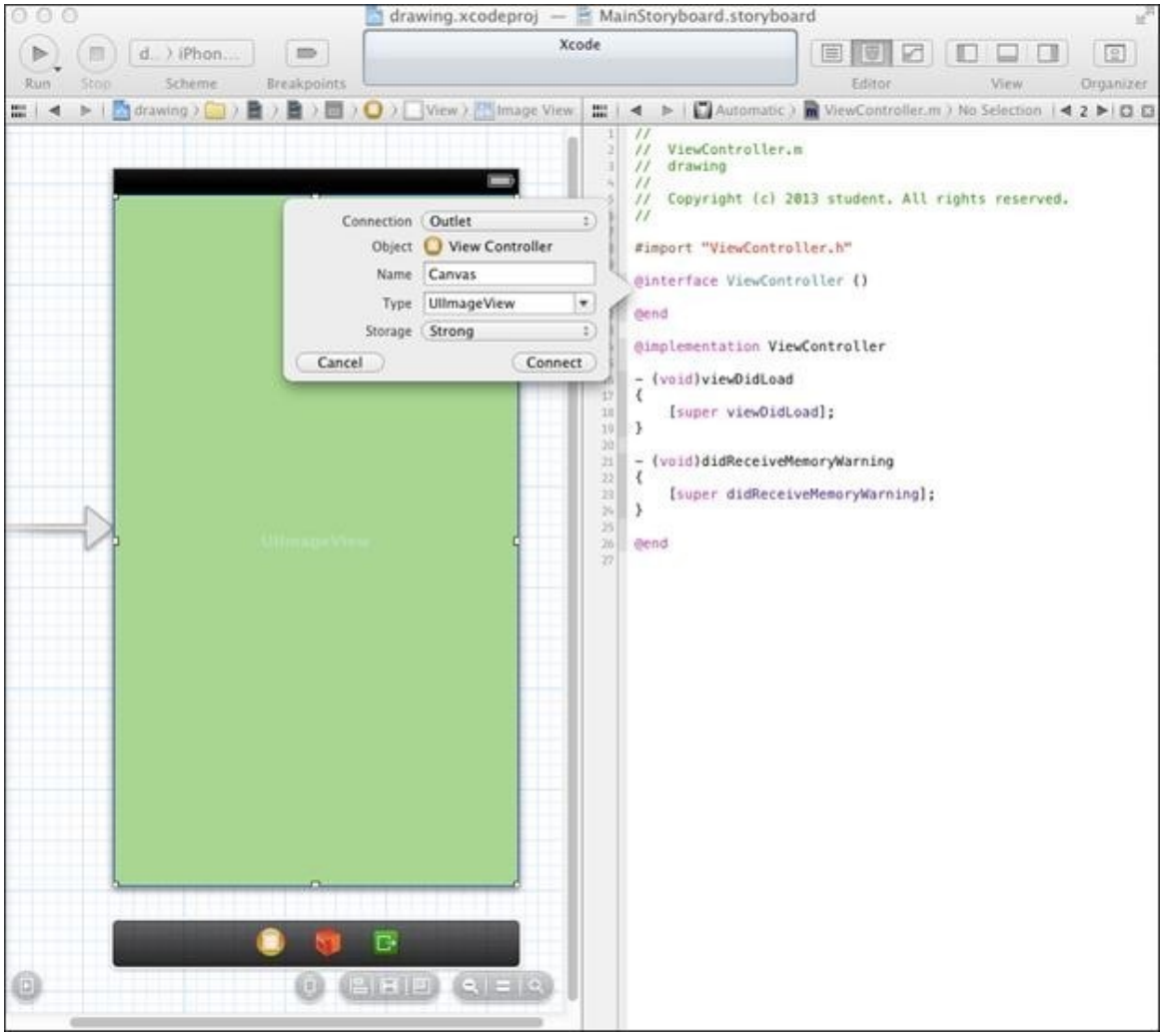

Рис. 3. Создание IBOutlet

Рисование будет осуществляться при помощи обработки жестов. Для этого в части implementation объявить методы отвечающие за обработку жестов (Листинг 2).

```
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event;

- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event;
```

Листинг 2. Объявление методов обработки жестов

Для рисования линии необходимо хранить предыдущую точку маршрута. Добавим свойство lastPont типа CGPoint в интерфейс контролера. CGPoint это обычная Си-шная структура хранящая координаты x и y (Листинг 3).

```
@property CGPoint lastPoint;
```

Листинг 3. Объявление свойства последней точки касания

Метод touchesBegan:withEvent: будет отвечать за сохранение начальной точки рисуемой линии в свойстве lastPoint (Листинг 4).

```
UITouch *touch = [touches anyObject];
```

```
[self setLastPoint:[touch locationInView:self.view]];
```

Листинг 4. Сохранение позиции касания

Метод touchesMoved:touchesMoved: будет отвечать за рисование линии на экране. Для рисования необходимо установить контекст рисования, установить параметры линии такие как цвет и размер, нарисовать линию и удалить контекст (Листинг 5).

```
UITouch *touch = [touches anyObject];

CGPoint currentPoint = [touch locationInView:self.view];

UIGraphicsBeginImageContext(self.view.frame.size);

CGRect drawRect = CGRectMake(0.0f, 0.0f, self.view.frame.size.width,
self.view.frame.size.height);

[[[self canvas] image] drawInRect:drawRect];

CGContextSetLineCap(UIGraphicsGetCurrentContext(), kCGLineCapRound);

CGContextSetLineWidth(UIGraphicsGetCurrentContext(), 5.0f);

CGContextSetRGBStrokeColor(UIGraphicsGetCurrentContext(), 1.0f, 0.0f, 0.0f, 1.0f);

CGContextBeginPath(UIGraphicsGetCurrentContext());

CGContextMoveToPoint(UIGraphicsGetCurrentContext(), _lastPoint.x,

_lastPoint.y);

    CGContextAddLineToPoint(UIGraphicsGetCurrentContext(),    currentPoint.x,
currentPoint.y);

CGContextStrokePath(UIGraphicsGetCurrentContext());

[[self canvas] setImage:UIGraphicsGetImageFromCurrentImageContext()];

UIGraphicsEndImageContext();

_lastPoint = currentPoint;
```

Листинг 5. Рисование линии на экране

1.2. Создание iOS приложения с простыми графическими объектами на языке Swift

Цель: изучить примеры рисования графических объектов и получить навыки разработки приложений

1. Просмотреть видео: https://www.youtube.com/watch?v=wmfecAPy_SM
2. Изучить статьи:
   - http://sketchytech.blogspot.com.by/2016/03/swift-giving-context-to-cgcontext-part-i.html
   - https://www.raywenderlich.com/128614/core-graphics-os-x-tutorial
   - https://www.raywenderlich.com/162315/core-graphics-tutorial-part-1-getting-started

- https://www.raywenderlich.com/162313/core-graphics-tutorial-part-2-gradients-contexts
- https://www.raywenderlich.com/167352/core-graphics-tutorial-part-3-patterns-playgrounds
- http://proswift.ru/calayer-ili-kak-zakruglit-ugly-sdelat-ten-i-gradient-na-swift/
- https://developer.apple.com/library/content/documentation/GraphicsImaging/Conceptual/drawingwithquartz2d/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007533-SW1

1.3. Создание iOS приложения на языке Swift с использованием анимации

Цель: изучить примеры рисования графических объектов и получить навыки разработки приложений.

1. Изучить статьи:
   - https://www.raywenderlich.com/173544/ios-animation-tutorial-getting-started-3
   - https://www.raywenderlich.com/173576/ios-animation-tutorial-custom-view-controller-presentation-transitions-3
   - https://developer.apple.com/library/content/documentation/WindowsViews/Conceptual/ViewPG_iPhoneOS/AnimatingViews/AnimatingViews.html
   - https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreAnimation_guide/CoreAnimationBasics/CoreAnimationBasics.html#//apple_ref/doc/uid/TP40004514-CH2-SW3

1.4. Создание iOS приложения на языке Swift с использованием жестов

Цель работы: Научиться обрабатывать события, которые будут происходить при использовании различных жестов.

Создать объект UILabel, который будет отображать используемый пользователем жест и менять цвет своего фона, в зависимости от используемого жеста. Реализовать работу следующих жестов:

1. Вращение (цвет фона: синий)
2. Масштабирование (цвет фона: красный)
3. Касание (цвет фона: зеленый)
4. Долгое нажатие (цвет фона: оранжевый)
5. Смахивание (цвет фона: серый)

Ход работы:

Создание проекта Xcode

Создаем проект Xcode, аналогично предыдущим лабораторным работам. Убираем галочку Use Core Data и выбираем iOS – Application – Single View Application.

Добавление элементов интерфейса

Открываем файл Main.storyboard и на экран контроллера View добавляем объект Label. Располагаем его посередине экрана и растягиваем (рис.11).

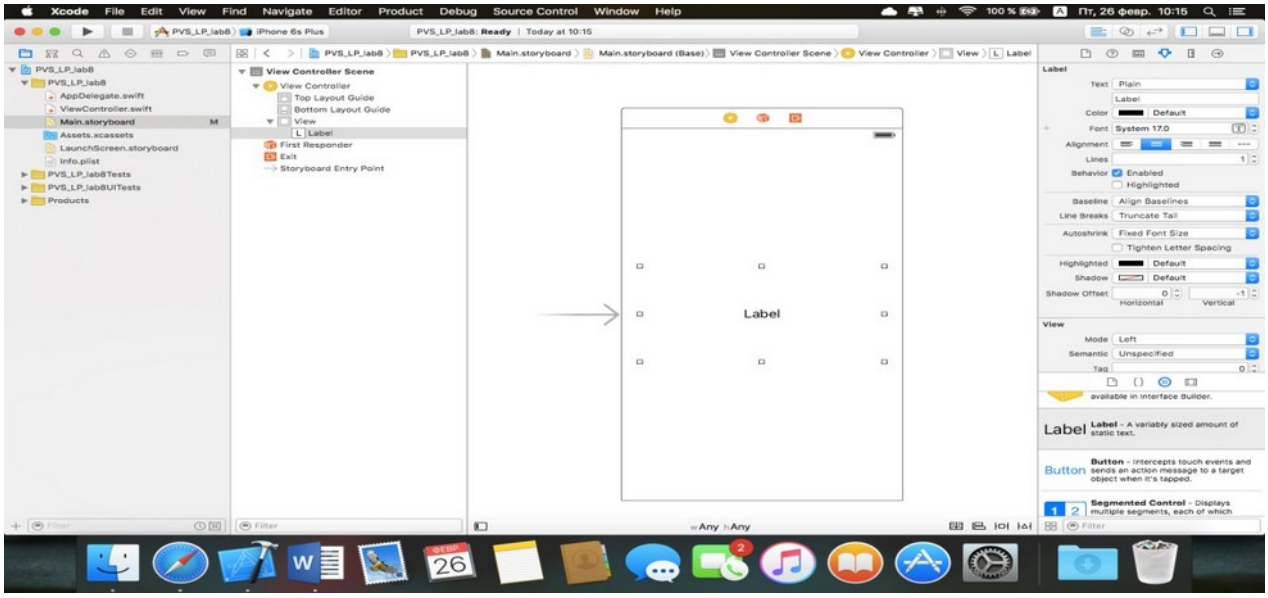

Рис. 11. Добавление объекта Label

Далее, находим объекты: Tap Gesture Recognizer, Long Press Gesture Recognizer, Pinch Gesture Recognizer, Rotation Gesture Recognizer и Swipe Gesture Recognizer. Перетаскиваем их в область объекта Label (рис. 12).

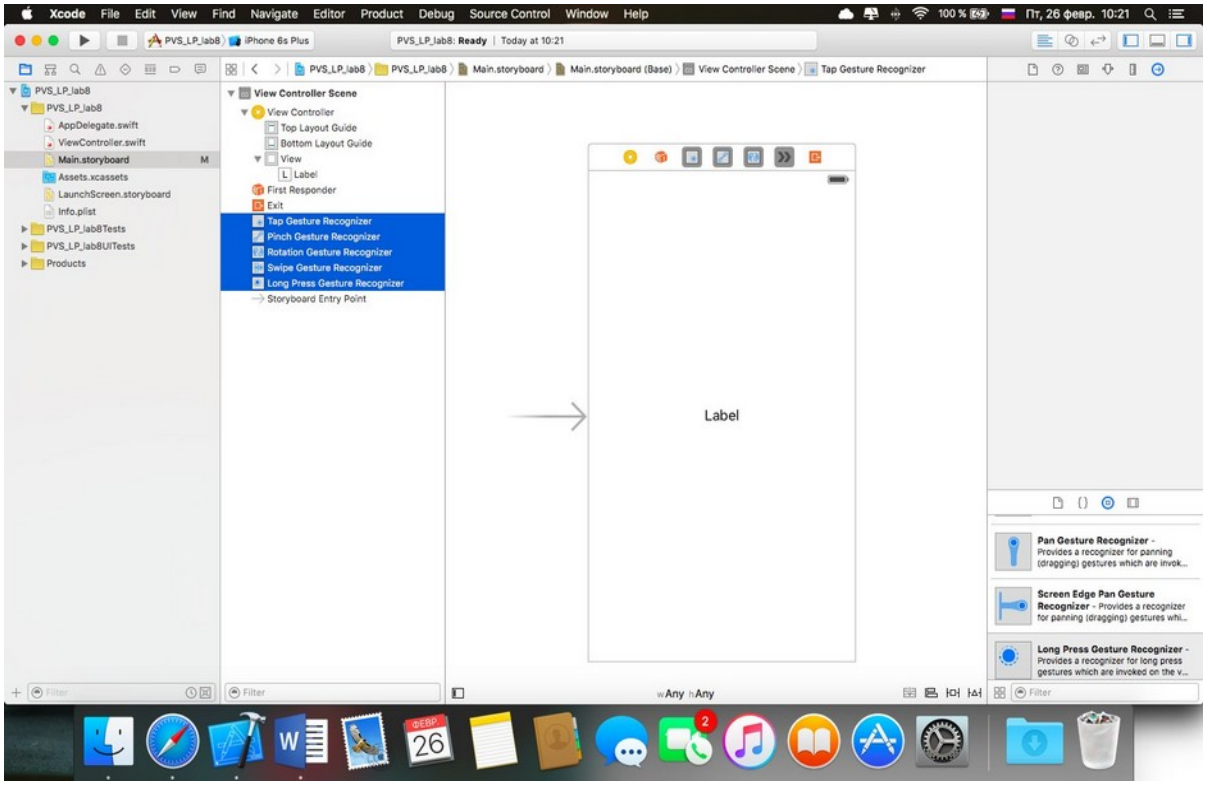

Рис.12. Область объекта Label

Создание связей объектов интерфейса с кодом приложения

В описание класса контроллера View для объекта Label добавляем связь типа Outlet:

```
@IBOutlet weak var gestureIndicator: UILabel!
```

Для объектов Gesture Recognizer добавляем пять связей типа Action:

```
@IBAction func tap(sender: AnyObject)
    {
```

```
    }
@IBAction func pinch(sender: AnyObject)
{

  }

@IBAction func rotation(sender: AnyObject)
{

}

@IBAction func swipe(sender: AnyObject)
{

    }

    @IBAction func longPress(sender: AnyObject)
    {

                                        }
```

Описание событий в коде приложения

Для того, чтобы использование жестов в области объекта Label стало возможным, присвоим параметру userInteractionEnabled (взаимодействие с пользователем включено) значение true в методе ViewDidLoad(): gestureIndicator.userInteractionEnabled = true

В этом же методе, задаем выравнивание текста, количество строк текста, текст и цвет фона:

gestureIndicator.textAlignment = NSTextAlignment.Center gestureIndicator.numberOfLines = 2 gestureIndicator.text = "Используйте жесты в этой области" gestureIndicator.backgroundColor = UIColor.yellowColor()

Далее, опишем события для всех пяти жестов в созданных ранее функциях. Для жеста касания:

```
        @IBAction func tap(sender: AnyObject)
    {
        gestureIndicator.text = "Жест: касание\n Цвет фона: зеленый"
    gestureIndicator.backgroundColor = UIColor.greenColor()    }
```

Для жеста масштабирования:

```
    @IBAction func pinch(sender: AnyObject)
    {
        gestureIndicator.text = "Жест: масштабирование\n Цвет фона: красный"
        gestureIndicator.backgroundColor = UIColor.redColor()
    }
```

Для жеста вращения:

```
    @IBAction func rotation(sender: AnyObject)
    {
```

```
        gestureIndicator.text = "Жест: вращение\n Цвет фона: синий"
    gestureIndicator.backgroundColor = UIColor.blueColor()
    }
        Для жеста смахивания:
        @IBAction func swipe(sender: AnyObject)
    {
        gestureIndicator.text = "Жест: смахивание\n Цвет фона: серый"
    gestureIndicator.backgroundColor = UIColor.lightGrayColor()
    }
        Для жеста долгого нажатия:
        @IBAction func longPress(sender: AnyObject)
    {
        gestureIndicator.text = "Жест: долгое нажатие\n Цвет фона: оранжевый"
        gestureIndicator.backgroundColor = UIColor.orangeColor()
    }
```

Сборка проекта и запуск приложения

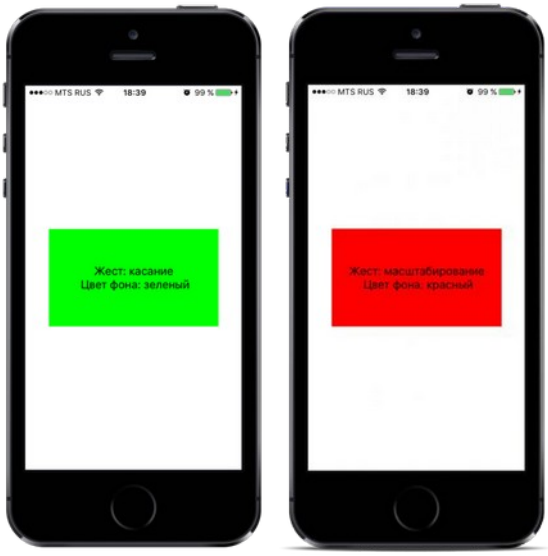

Рис. 13.

2. ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

ВАРИАНТЫ ЗАДАНИЙ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Список вариантов для заданий 2.2-2.4:

1. Пятиконечная звезда и круг
2. Пятиугольник и шестилистник
3. Криволинейный треугольник и шестиугольник
4. Четырехлистник и параллелограм
5. Разносторонний тупоугольный треугольник и десятиугольник
6. Восьмилистник и равнобедренный треугольник
7. Пятилистник и равносторонняя трапеция
8. Четырехугольник и девятилучевая звезда
9. Октаграмма (восьмилучевая звезда) и окружность

10. Гексаграмма (шестиконечная звезда) и ромб
11. Четырехлистник и десятиугольник
12. Криволинейный треугольник и окружность
13. Выпуклый дельтоид и разносторонний тупоугольный треугольник
14. Семиугольник и правильная четырехконечная звезда
15. Восьмиугольник и тупоугольная трапеция
16. Декагон и равнобедренная трапеция
17. Круговой сегмент и прямоугольная трапеция
18. Круговой сектор и тупоугольный треугольник
19. Невыпуклый (вогнутый) дельтоид и семиугольник
20. Разносторонний тупоугольный треугольник и квадрат
21. Трилистник и круг
22. Шестилистник и выпуклый дельтоид
23. Ромб и круговой сегмент
24. Октаграмма и квадрат
25. Тупоугольная трапеция и правильная пятиконечная звезда
26. Выпуклый дельтоид и равнобедренная трапеция
27. Трилистник и пятиугольник
28. Октаграмма (восьмилучевая звезда) и параллелограмм
29. Криволинейный треугольник и параллелограмм.
30. Правильная четырехконечная звезда и прямоугольник.
31. Вогнутый дельтоид и шестиугольник.
32. Октаграмма и равносторонний треугольник.
33. Правильная пятиконечная звезда и тупоугольный треугольник.
34. Равнобедренная трапеция и круговой сектор.
35. Восьмилистник и квадрат.
36. Десятиугольник и сегмент.
37. Девятилучевая звезда и трапеция.
38. Гексаграмма и равносторонний треугольник.
39. Пятилистник и ромб.
40. Шестиугольник и сегмент.

2.1 Рисование (приложение на языке objective-C)

Дополнить функционал приложения, созданного в 1.1. Требуется разработать программу рисования. Интерфейс программы должен поддерживать рисование линиями 5 размеров и разными цветами.

Реализовать возможность сохранения нарисованного изображения в файл.

Исходный код приложений на языке Objective-C должен соответствовать модели КИС для Objective-C.

Входные данные

Цвет, размер и форма кисти, использующейся для рисования.

Выходные данные

Рисунок, состоящий из линий произвольной формы, сохранённый в библиотеку.

2.2. Создание iOS приложения с простыми графическими объектами на языке Swift

Создать iOS приложение, иллюстрирующее объекты двумерной графики с однотонным фоном, градиентом и тенью, согласно вариантам. Проиллюстрировать различные варианты создания новых фигур за счет вычитания или объединения 2 разных фигур.

Код проекта распределить по группам (каталогам) согласно концепции MVC (Model, View, Controller).

2.3. Создание iOS приложения с использованием анимации

Дополнить iOS приложение, созданное в задании 2.2, анимационными эффектами. Проиллюстрировать следующие эффекты:

- Перемещение
- Вращение
- Масштабирование
- Прозрачность
- Наложение как минимум двух эффектов, например перемещение и вращение

Код проекта распределить по группам (каталогам) согласно концепции MVC (Model, View, Controller).

2.4. Создание iOS приложения на языке Swift с использованием жестов

Изменять **фон одной из геометрических фигур**, созданных в приложении в задании 2.2, в зависимости от жеста. Проиллюстрировать следующие жесты:

1. Вращение (у фигуры устанавливается фон 1)
2. Масштабирование (у фигуры устанавливается фон 2)
3. Касание (у фигуры устанавливается фон 3)
4. Долгое нажатие (у фигуры устанавливается фон 4)
5. Смахивание (у фигуры устанавливается фон 5)

Для выполнения задания подобрать 5 изображений фона. Код проекта распределить по группам (каталогам) согласно концепции MVC (Model, View, Controller).

3. ЗАДАНИЕ ПОВЫШЕННОГО УРОВНЯ

3.1 Рисование 3D-графики на языке Swift

1) В случае разработки проекта на основе библиотеки Metal изучить документацию и пример по созданию проекта с использованием Metal, который выполняется в симуляторе:

- https://developer.apple.com/documentation/metal/developing_metal_apps_that_run_in_simulator
- скачать пример проекта https://developer.apple.com/documentation/metal/supporting_simulator_in_a_metal_app

2) Изучить документацию по OpenGL ES или Metal

OpenGL ES: Разобрать пример из статьи https://www.raywenderlich.com/5146-glkit-tutorial-for-ios-getting-started-with-opengl-es и документацию https://developer.apple.com/library/archive/documentation/3DDrawing/Conceptual/OpenGLES_ProgrammingGuide/Introduction/Introduction.html#//apple_ref/doc/uid/TP40008793

Metal: Разобрать пример из статей:

- https://www.raywenderlich.com/7475-metal-tutorial-getting-started
- https://www.raywenderlich.com/976-ios-metal-tutorial-with-swift-part-5-switching-to-metalkit
- https://academy.realm.io/posts/3d-graphics-metal-swift/
- https://medium.com/@jscampbell/bare-metal-working-with-metal-and-the-simulator-70e085e3a45

2) Разработать простую трехмерную игру, используя технологию OpenGL ES и язык программирования GLSL или Metal. Трехмерная сцена должна поддерживать вращение и масштабирование объекта произвольной формы, например чайника/вазы или другого объекта, с помощью жестов.

Код проекта распределить по группам (каталогам) согласно концепции MVC (Model, View, Controller).

Входные данные

Массив вершин, хранящий позицию и цвет каждой вершины модели чайника/вазы или другого объекта.

Выходные данные

Изображение трёхмерной модели чайника/ кувшина / вазы / кольца.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1) Как нарисовать контур?
2) Как нарисовать прямоугольник?
3) Как заполнить контур цветом? Градиентом?
4) Как заполнить прямоугольник цветом, сохранив цвет контура?

5) Как нарисовать круг / эллипс?

6) Как добавить тень к рисункам?

7) Как нарисовать изображение на контроллере View в прямоугольнике?

8) Как перерисовать представление View с помощью метода SetNeedsDisplay

9) Как вычесть одну фигуру из другой?

10) Какое компонент Cocoa отвечает за обработку жестов?

11) Как добавить и обработать жест касания (нажатия) (tap gesture)?

12) Как добавить и обработать долгое нажатие (long press gesture)?

13) Как добавить и обработать жест перелистывания (смахивания) (swipe gesture)?

14) Как обрабатывается жест стягивания (щипка) (pinch gesture)?

15) Как обрабатывается жест растягивания (spread gesture)?

16) Как добавить жесты непосредственно на Storyboard?

17) Какие примитивы рисования Вы знаете?

18) Что такое CGContext?

19) Что такое UIBezierPath? Чем отличается от CGContext?

20) Что такое CGImage?

21) Что такое CGPath?

22) В чем заключаются трудности использования двумерной графики?

23) Что такое OpenGL?

24) Что такое шейдер?

25) Чем вершинный шейдер отличается от пиксельного?

26) Что такое Metal?