Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский государственный университет»

Кафедра системного программирования

Гудиев Артур Владимирович

Реализация примитивов и оконного менеджера для построения пользовательских интерфейсов на языке PostScript

Выпускная квалификационная работа специалиста

Допущена к защите.
Зав. кафедрой:
д.ф.-м.н., проф. А.Н. Терехов

Научный руководитель:
к.ф.-м.н. Д.Ю. Булычев

Рецензент:
к.ф.-м.н., доц. Д.В. Кознов

Санкт-Петербург

2015

SAINT-PETERSBURG STATE UNIVERSITY
Mathematics & Mechanics Faculty
Department of Software Engineering

Artur Gudiev

Window manager and GUI primitives for user interface implementation in PostScript

Graduation Thesis

Adnitted for defence.
Head of the chair:
professor Andrey Terekhov

Scientific supervisor:
PhD, Acc. Prof. Dmitri Boulytchev

Reviewer:
PhD, Acc. Prof. Dmitri Koznov

Saint-Petersburg

2015

Оглавление

Введение

PostScript - это графический интерпретируемый язык программирования, созданный с целью представления графики (текстовых фалов, рисунков, чертежей) в машинонезависимой форме [1]. С помощью графических операторов языка PostScript можно определить область рисования, отобразить прямые и кривые линии, залить цветом область, задать различные графические параметры.

Графический интерфейс пользователя является разновидностью пользовательского интерфейса и состоит из различных примитивов - окон, полей ввода, кнопок и т.д. Язык PostScript обладает базовыми возможностями для реализации внешнего вида графических примитивов.

Для полноценной работы графических интерфейсов реализации одних примитивов недостаточно. Требуется также оконный менеджер — приложение, управляющее размещением примитивов и определяющее их внешний вид. Оконный менеджер позволяет добавлять и удалять примитивы, управлять порядком отображения, пересчитывать координаты и т.д. В оконном менеджере могут быть реализованы также и визуальные эффекты, проявляющиеся во время работы с окнами (например, эффект волны и эффект упорядочивания окон). В каждой операционной системе существует свой оконный менеджер, который взаимодействует с графическими интерфейсами, созданными с помощью, например, Qt [5] или Swing [6].

Ранее в рамках проекта лаборатории JetBrains был реализован интерпретатор PostScript [2] [3] [4]. Однако с его помощью было трудно создать графические интерфейсы, а также реализовать оконный менеджер, так как, например, в PostScript не поддерживается механизм обработки событий. Для упрощения реализации этой возможности было решено расширить язык PostScript и на

его основе разработать графическую библиотеку, позволяющую создавать графические интерфейсы. Данную работу можно разделить на три направления: оптимизация интерпретатора (Д. Поздин), обработка событий (Р. Макулов) и реализация графических примитивов и оконного менеджера (А. Гудиев). То, что интерпретатор PostScript реализован на языке Java, делает создаваемые им интерфейсы кроссплатформенными.

Целью данной дипломной работы является реализация графических примитивов и оконного менеджера на языке PostScript для решения задачи реализации кроссплатформенных пользовательских интерфейсов.

1 Обзор

1.1 Описание существующих решений

Qt Quick

Qt — это кроссплатформенный фреймворк для создания программного обеспечения, написанный на языке C++. Qt Quick является средой разработки пользовательских интерфейсов, расспространяемой вместе с Qt. Qt Quick состоит из декларативного языка QML и стандартной библиотеки компонентов QtQuick. В Qt Quick реализована новая концепция логики моделирования приложений, основанная на иерархической машине состояний [5]. Кроме того, богатый набор анимаций (tweens) облегчают создание полнофункциональных пользовательских интерфейсов.

Как предметно-ориентированная среда разработки она специально ориентирована на создание пользовательских интерфейсов. В Qt Quick нет проблем с приведением типов, указателями и временем жизни объектов. Вместо этого, в центре внимания находится создание богатых пользовательских интерфейсов.

К недостаткам Qt Quick можно отнести необходимость перекомпиляции одной программы для разных платформ.

Swing

Компания Sun разработала набор графических компонентов под названием Swing [6]. Компоненты Swing полностью написаны на Java. Для отрисовки используется 2D - API для рисования двумерной графики на языке Java. В Swing легко создавать новые компоненты.

Благодаря простоте использования, детальной документации и гибкости архитектуры Swing стал, пожалуй, самым популярным графическим фреймвор-

ком для Java. На его основе создано появилось много расширений, таких как SwingX [7], JGoodies [8], которые значительно упрощают создание сложных пользовательских интерфейсов. Практически все популярные среды программирования Java включают графические редакторы для Swing-форм, что облегчает освоение Swing.

Особенность Swing заключается в том, что приложения Swing могут выглядеть одинаково и по-разному на различных платформах. Приложения, которые внешне одинаково отображаются в разных операционных системах, относительно медленно работают.

1.2 Описание используемых инструментов

JVM

Виртуальная машина Java (сокращенно JVM) - основная часть среды выполнения для Java, так называемой Java Runtime Environment (JRE). Программы на языке Java компилятором Java (javac) переводятся в байт-код Java. Виртуальная машина Java исполняет байт-код. В настоящее время JVM получила широкое распространение. Ее реализация есть для многих операционных систем, что делает программы, написанные на Java, кроссплатформенными. Интерпретатор PostScript, реализованный в рамках проекта JetBrains, написан на языке Java.

Swing и AWT

Swing и AWT — это библиотеки языка Java, с помощью которых в реализованном интерпретаторе строится путь рисования, отображаются шрифты, задаются параметры графической среды исполнения такие, как цвет, ширина линии, способы соединения отрезков и рисования концов отрезка.

Интерпретатор PostScript

Интерпретатор PostScript, реализованный в рамках проекта JetBrains, написан на Java. Он состоит из следующих пакетов: парсер (parser), среда исполнения (runtime) и интерпретатор (interpreter).

Парсер разбивает входной поток на токены согласно синтаксису PostScript и записывает их в виде процедуры.

Среда исполнения содержит в себе реализацию основных операторов (таких как add, if, for, show ...) и объектов, то есть их значений, типов и атрибутов. Среда исполнения содержит также и класс AVL-дерева, используемый в словарях, стеки, виртуальную память, в которой хранятся значения сложных объектов и графическую среду исполнения. В графической среде исполнения описаны классы отображения результата программы на экране, матрица преобразования системы координат, класс пути, с помощью которого строится изображение, класс снимка графического состояния, используемый при сохранении или восстановлении графического состояния и набор графических параметров (таких как цвет, ширина линии, форма концов отрезка ...).

Интерпретатор передает парсеру программу PostScript, получает в качестве результата процедуру, кладет ее на стек исполнения и запускает выполнение стека исполнения.

1.3 Проект по разработке графической библиотеки PostScript

Задача реализации интерпретатора графического языка PostScript решается в рамках проекта лаборатории JetBrains. Одно из назначений библиотеки — создание кроссплатформенных пользовательских интерфейсов.

Текущую деятельность в проекте можно разделить на следующие задачи:

- оптимизация интерпретатора;

- добавление механизма обработки событий;
- реализация примитивов и оконного менеджера.

В языке PostScript нет механизма обработки событий, что осложняет создание интерфейсов. Поэтому было решено расширить язык PostScript, добавив в него новые операторы, и, используя средства языка Java, реализовать поддержку событий. Используя механизм событий, требуется реализовать графические примитивы (окно, поле ввода, кнопка и т.д.) и оконный менеджер. Примитивы должны отображаться на экране средствами языка PoscScript. Кроме того, необходимо, чтобы при возникновении событий у примитивов исполнялись заданные процедуры. Требуется также, чтобы оконный менеджер управлял порядком отображения примитивов, их добавлением и удалением, пересчетом координат, перерисовкой и т.д. Использование в программе механизма обработки событий, графических примитивов и оконного менеджера может существенно замедлить исполнение программы, поэтому для полноценной работы создаваемых интерфейсов необходима оптимизация интерпретатора.

2 Реализация графических примитивов

В данной главе описывается сначала общая внутренняя структура примитивов, а затем особенности каждого примитива.

2.1 Структура примитивов

Код графческой библиотеки языка PostScript содержится в файле glib.ps. В нем создается словарь gelements. У каждого примитива есть свой номер, по которому он хранится в словаре gelements.

У графических примитивов реализовано отношение наследования. Общий предок у всех — объект сцена (scene). У каждого примитива есть дети-примитивы. Номера детей хранятся в поле-массиве children.

Каждому графическому примитиву соответствуют два файла — файл с описанием объекта и файл с процедурой отрисовки примитива (например, для кнопки — это button.ps и paintButton.ps).

В файле первом есть два конструктора — абсолютный и относительный. У каждого примитива есть свои координаты и размеры. Если координаты задаются относительно родителя, то вызывается относительный контсркутор. Если примитив задается абсолютными координатами, тогда вызывается абсолютный. Например, пусть есть окно с левым нижним концом в точке (0,0), высотой 1000 и шириной 1000. Пусть от окна наследуется кнопка с левым нижним концом в (200, 300), длиной 400 и шириной 500. Тогда абсолютный контсруктор будет вызываться с параметрами 200, 300, 400 и 500. А относительный — с 0.2, 0.3, 0.4 и 0.5. Относительные координаты полезны тем, что при изменении размеров или сдвиге родителя, они не изменяются.

Каждому примитиву можно добавить событие - процедуру PostScript. Их может быть несколько. У каждого графического объекта есть словарь eventProcDict,

в нем хранятся процедуры, которые вызываются при событии у объекта.

2.2 Примитивы графической библиотеки

Кнопка — это примитив, у которого вызывается заданное событие по нажатию. У кнопки есть два варианта отрисовки: с обычной и с нажатой кнопкой. В процедуре рисования проверяется, нажата ли кнопка. Если да, то рисуется затемненная кнопка без тени (см. рис. 2), а иначе светлая с тенью (см. рис. 1) .

Рис. 1: Кнопка

Рис. 2: Нажатая кнопка

Примитив **флажок** позволяет вызывать заданные процедуры при снятии и задании флага. В процедуре отрисовки проверяется, стоит ли флаг. Если стоит, то рисуется галочка в закрашенной области (см. рис. 4), а иначе отображается пустая область (см. рис. 3).

Рис. 3: Флажок

Рис. 4: Отмеченный флажок

Примитив **поле со списком** позволяет выбрать один из заданных вариантов. Данный примитив состоит из поля, в котором отображается выбранный вариант, и списка со всеми вариантами. Изначально список свернут, отображается только поле (см. рис. 5). При нажатии на поле список раскрывается, отображаются все варианты (см. рис. 6). При следующем нажатии по координатам клика проверяется, был ли выбран какой-нибудь вариант, и если да, то вычисляется, какой именно. После этого список сворачивается, и в поле отображается новый вариант.

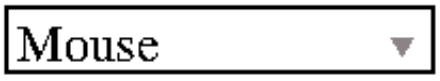

Рис. 5: Поле со списком

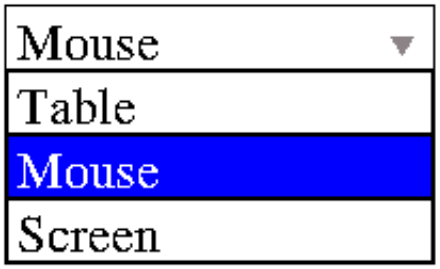

Рис. 6: Раскрытое поле со списком

Список также предоставляет возможность выбора одного из имеющихся вариантов. Отличие списка от поля со списком в том, что в первом список всегда развернут, то есть все варианты постоянно отображаются (см. рис. 7). Выбранный вариант рисуется на синем фоне. При нажатии на список проверяется, на каком варианте произошел клик. Затем старый выбранный вариант уже отображается на белом фоне, а новый выбранный - на синем.

Метка - это примитив, который отображает текст на экране (см. рис. 8). При необходимости можно самостоятельно задать размер шрифта и цвет. Для этого нужно в словаре метки поставить по ключу isAttached значение true, и задать необходимые значения ключам kegel и color.

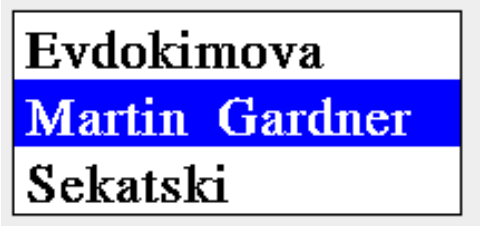

Рис. 7: Список

Рис. 8: Метка

Примитив **поле редактирования** предназначен для ввода текста. Изначально в поле ничего не отображается. При нажатии на поле, появляется курсор. Сразу после ввода символы появляются на экране. Курсор можно сдвигать влево и вправо, в начало и конец строки. Напечатанные символы можно удалять. Если введенная строка превышает ширину поля, то вычисляется, какая подстрока отображается на экране (см. рис. 9).

Рис. 9: Поле редактирования

Радиокнопка - это примитив, который может находиться в включенном или выключенном состоянии. При отрисовке радиокнопки проверяется, в каком она состоянии, и в зависимости от этого отображается включенная (см. рис. 11) или выключенная радиокнопка (см. рис. 10).

Рис. 10: Включенная радиокнопка

Рис. 11: Выключенная радиокнопка

Окно позволяет хранить и отображать в себе другие примитивы (см. рис. 12). У окна можно динамически изменять размеры. Для этого нужно подвести курсор к границе окна, курсор при этом должен поменяться, затем нажать и расстянуть или сжать окно до нужных размеров. При этом у окон задан минимальный размер, меньше которого сжать не получится. Для изменения курсора был введен новый оператор cursor. Окно также можно и перемещать. Для этого нужно зажать верхнюю область окна, и перенести курсор вместе с окном в нужное место. К окну автоматически добавляются служебные примитивы: метка-надпись в верхней части окна и кнопка закрытия. Кнопка закрытия удаляет из словаря графических притивов gelements само окно и всех его потомков.

Рис. 12: Окно

3 Оконный менеджер

В данной главе описаны компоненты, из которых состоит оконный менеджер. Оконный менеджер содержит в себе набор процедур, отвечающих за добавление и удаление примитивов, порядок отображения окон, перемещение и изменение размеров окон, пересчет координат, перерисовку примитивов и визуальные эффекты (см. рис. 13).

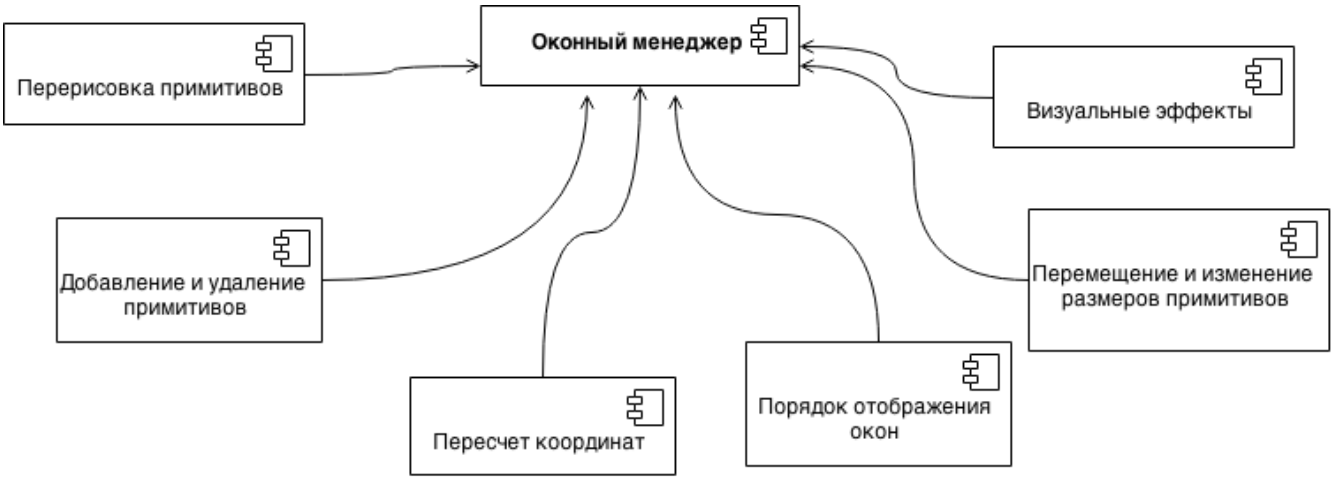

Рис. 13: Оконный менеджер

3.1 Добавление и удаление примитивов

Для создания примитива нужно вызвать его конструктор с необходимыми параметрами (расположение, размеры, события...). Конструктор создает словарь, в котором хранятся параметры примитива, и вызывает процедуру addElement, которая добавлет примитив-словарь в словарь gelements. Для удаления примитва предназначены продцедуры deleteWithChildren и deleteFromParent. deleteWithChildren удаляет сам примитив и всех его потомков из словаря графических примитивов gelements, а deleteFromParent удаляет

номер примитива в списке детей у родителя.

3.2 Порядок отображения окон

В программе может быть несколько окон, перекрывающих друг друга. Изначально все окна отображаются в том порядке, в каком они создавались. Но далее, если нажать на произвольное окно, то оно оказывается *в фокусе*. В графической библиотеке glib.ps в словаре gelements создается элемент focusedElement. Изначально это сцена. Но потом при нажатии на произвольный примитив, focusedElement становится равным выбранному примитиву. Если на окно нажали, то оно помимо того, что становится в фокусе, ещё и в списке детей родителя ставится последним. Это означает, что оно будет отображаться поверх остальных детей родителя, так как они рисуются в порядке списка.

3.3 Перемещение и изменение размеров окон

Для перемещения и изменения размеров окон в графической библиотеке glib.ps в словаре gelements создаются элементы resizingElement и draggingElement, отвечающие соответственно за изменение размеров окна и его перемещение. Изначально они равны null. При нажатии на окно проверяется область нажатия. Если нажата граничная область окна, то resizingElement становится равным данному окну, иначе если нажата верхняя часть окна, то draggingElement оказывается равным нажатому окну. При отпускании окна resizingElement и draggingElement вновь обнуляются.

3.4 Пересчет координат

При перемещении или изменении размеров окна, у примитвов-потомков изменяются абсолютные координаты, поэтому их необходимо пересчитывать. За пересчет координат отвечает процедура refreshCoordinatesForChildren. Она у всех детей окна вызывает процедуру refreshCoordinates, которая по своим относительным координатам и абсолютным координатам родителя пересчитывает свои абсолютные координаты.

3.5 Перерисовка примитивов

При некоторых событиях изменяется внешний вид примитивов, поэтому нужна перериосвка. За перерисовку отвечает процедура repaintAll. Она очищает экран с помощью оператора init и запускает с помощью процедуры paintChildren отрисовку всех примитивов, вызывает у каждого примитива процедуру paint.

3.6 Визуальные эффекты

К примитиву сцене можно добавить процедуры-события для отображения визуальных эффектов. В частности, сейчас добавлено событие волна: при нажатии правой кнопки мыши происходит эффект волны.

4 Демонстрационные примеры

В данной главе приводится описание трех демонстрационных примеров: формы, демонстрации эффектов и калькулятора. На данных примерах тестировались графические примитивы и оконный менеджер, а также их взаимодействие с другими компонентами проекта (например, с механизмом обработки событий).

4.1 Форма

Ниже приведен демонстрационный пример, содержащий в себе все реализованные примитивы (см. рис. 14). Здесь есть надписи "Welcome!"и "Fill in this form, please."За ними идут напдись "Your name"и поле для ввода своего имени. Затем метка "Group"и поле для ввода номера своей группы. После этого расположены надпись "Education"и список для выбора формы обучения. За формой обучения идут метка "Faculty"и полем со списком для выбора факультета.

Далее флажок с текстом "I want to get results"для подтверждения того, что пользователь хочет получить результат. Если флажок выбран, и пользователь, заполнив форму (см. рис. 15), нажимает кнопку закрытия "Close" , то данные с примитивов считываются, записываются в словарь, который затем кладется на стек операндов.

После этого идет надпись "Error checking"и радиокнопка с надписями "On" и "Off" . Если радиокнопка включена, то при выходе из программы проверяется корректность введенной группы, и если группа введена некорректно, то будет показано окно с ошибкой. Замыкает список примитивов кнопка "Close" . При условии успешного прохождения проверки ошибок появляется окно с вопросом, действительно ли пользователь хочет выйти. Если пользователь выбрал кнопку "no" , то окно с вопросом просто закрывается. Если же выбрана кнопка "yes" , то закрывается вся программа.

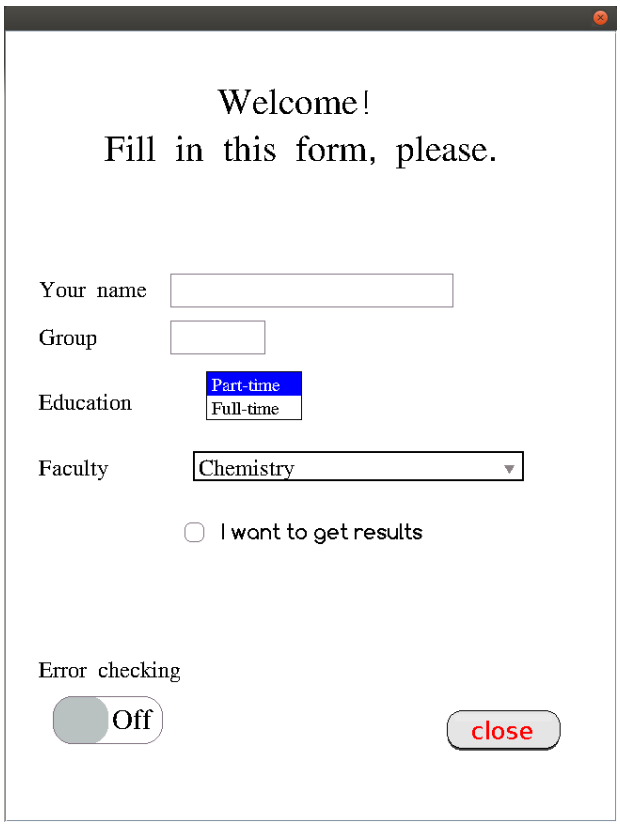

Рис. 14: Пустая форма

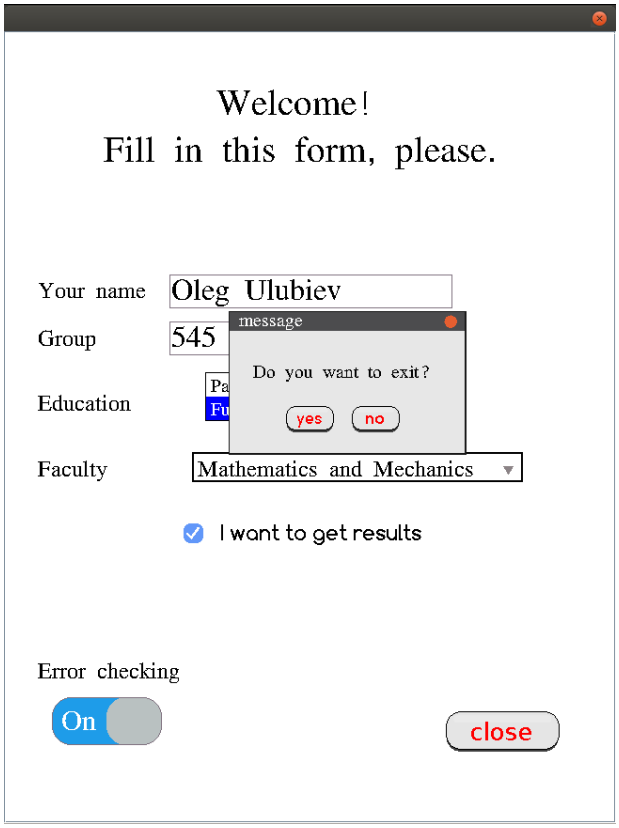

Рис. 15: Заполненная форма

Данный пример демонстрирует работу всех примитивов, а также возможность сохранения данных. Кроме того здесь есть создание нового примитива по событию.

4.2 Демонстрация эффектов

В данном демонстрационном примере показывается, как работают различные эффекты, прикрепленные к событиям. Сначала идут надписи "Hello!" и "Here you can see various effects". За ними идут надпись "scales" и кнопки увеличения и уменьшения масштаба. Затем метка "translates" и четыре кнопки перемещений. Далее расположены надпись "rotates" и две кнопки вращения, при нажатии которых происходит поворот (см. рис. 17). За формой обучения

идут метка "reflections" и два флажка, включающих эффект отражения. Затем следует надпись "invert labels" и кнопка с событием изменения надписей в программе. В конце расположена кнопка "Close", закрывающая программу. Кроме того, если нажать правой кнопкой мыши в области интерфейса, то появляется эффект волны (см. рис. 16).

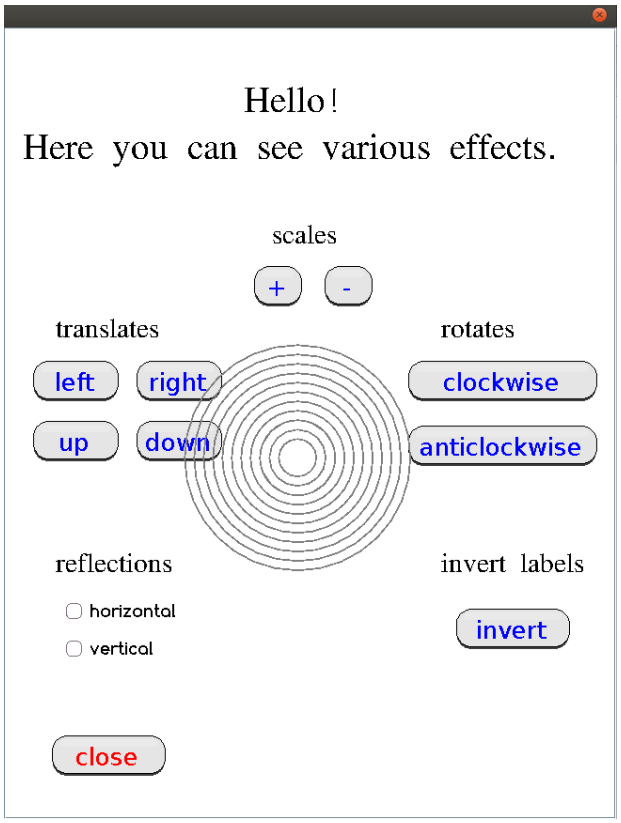

Рис. 16: Эффект волны

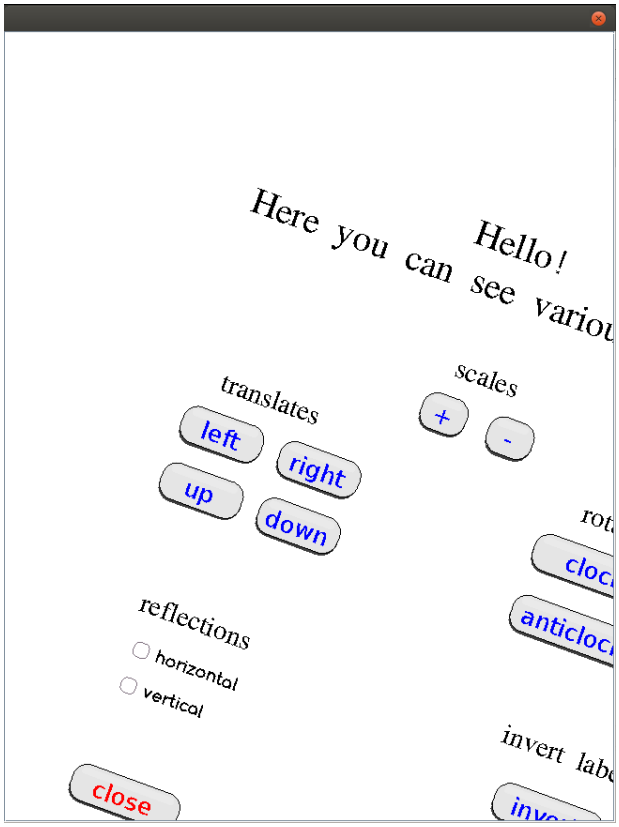

Рис. 17: Поворот

Данный пример демонстрирует работу различных преобразований системы координат, а также эффекта волны.

4.3 Калькулятор

Третий пример показывает, как можно не только считывать информацию с примитивов, но изменять их состояния по событиям других примитивов. Сна-

чала идут надписи "Welcome!" и "You can calculate here". Затем идет окно-калькулятор. В нем есть поле для ввода и отображения чисел, кнопка очищения поля, а также кнопки с цифрами, арифметическими операциями, точкой и знаком равно. Вычисления выполняется с помощью конечного автомата (см. рис. 18), в котором есть 4 состояния (1 — ввод первого числа, 2 — сохранение первого числа и операции, 3 — ввод второго числа, 4 — вывод результата). В программе создается словарь calculator и в нем хранятся состояние, операция и введенные числа.

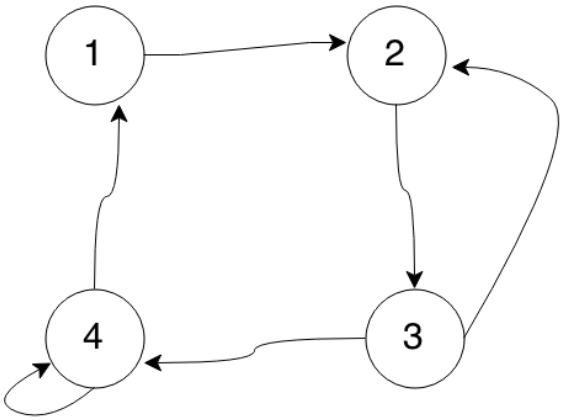

Рис. 18: Конечный автомат

Из третьего состояния можно вернуться во второе, если после ввода второго операнда выбирается новая арифметическая операция. При этом автоматически вычисляется и отображается результат старых операндов, результат сохраняется в качестве первого числа, а второе число не меняется.

Петля в четвертом состоянии показывает, что если несколько раз нажать знак равенства, то к результату каждый раз будет применяться последняя операция со вторым операндом.

При вычислении результата проверяется, какого тип результат (real или integer). Если это возможно, то идет приведение типов (приведение real к integer).

Кроме того, при выводе результата, вычисляется какая подстрока строки будет выведена на экран (с помощью процедуры setTextToTextField).

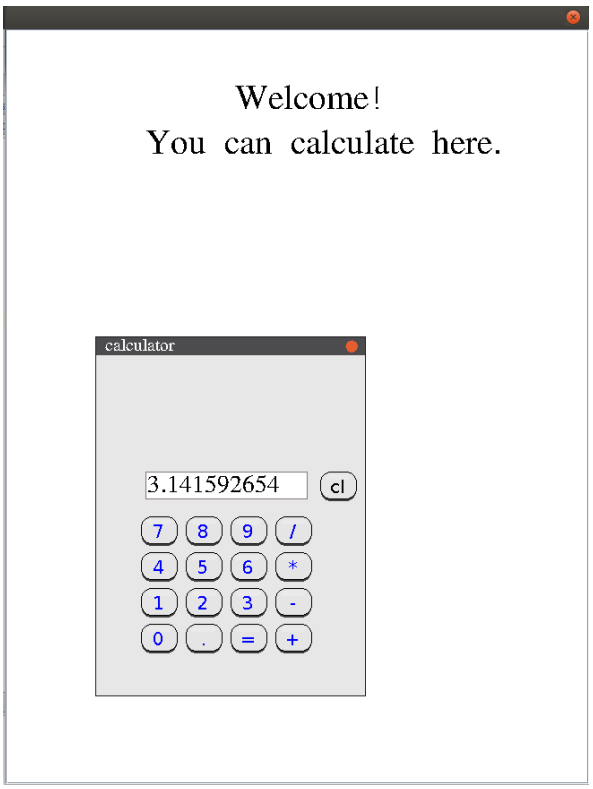

Рис. 19: Калькулятор

Данный пример демонстрирует сохранение данных и их дальнейшее использование в событиях (см. рис. 19).

Заключение

В рамках дипломной работы получены результаты, перечисленные ниже.

- Добавлены следующие примитивы в графическую библиотеку PostScript: кнопка, флажок, поле со списком, список, метка, поле редактирования, радиокнопка, окно.
- Реализован оконный менеджер, интегрированный с графической библиотекой.
- Проведено тестирование оконного менеджера на демонстрационном примере.

Список литературы

[1] Спецификация языка Postscript. PostScript Language reference.
Adobe Systems. 1999
http://www.adobe.com/products/postscript/pdfs/PLRM.pdf

[2] Дмитрий Поздин. Реализация общей поддержки времени исполнения для интерпретатора языка PostScript.// Труды лаборатории языковых инструментов. Выпуск 2. 2014. с. 276-296.

[3] Артур Гудиев. Реализация графической части интерпретатора языка PostScript.// Труды лаборатории языковых инструментов. Выпуск 2. 2014. с. 297-312.

[4] Рустам Макулов. Архитектура интерпретатора для исполнения программ на языке PostScript в JVM. // Труды лаборатории языковых инструментов. Выпуск 2. 2014. с. 259-275.

[5] [Qt Developer Network Wiki], https://wiki.qt.io/IntroductionQtQuick/ru

[6] [Хакер], https://xakep.ru/2014/09/10/java-gui/

[7] [The Source for Java Technology Collaboration], https://swingx.java.net/

[8] [JGoodies], http://www.jgoodies.com/