МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования   
**«Национальный исследовательский   
Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Отчет по практическому заданию для лекции №9**

**Выполнила:**

студентка группы 382006-2

Кулёва Анна Андреевна

**Проверил:**

Карчков Денис Александрович

Рецензент:

Нижний Новгород  
2023

# Содержание

[Содержание 2](#_Toc151220350)

[1. Цель практического занятия 3](#_Toc151220351)

[2. Постановка задачи 4](#_Toc151220352)

[3. Руководство пользователя 6](#_Toc151220353)

[4. Руководство программиста 10](#_Toc151220354)

[Заключение 12](#_Toc151220355)

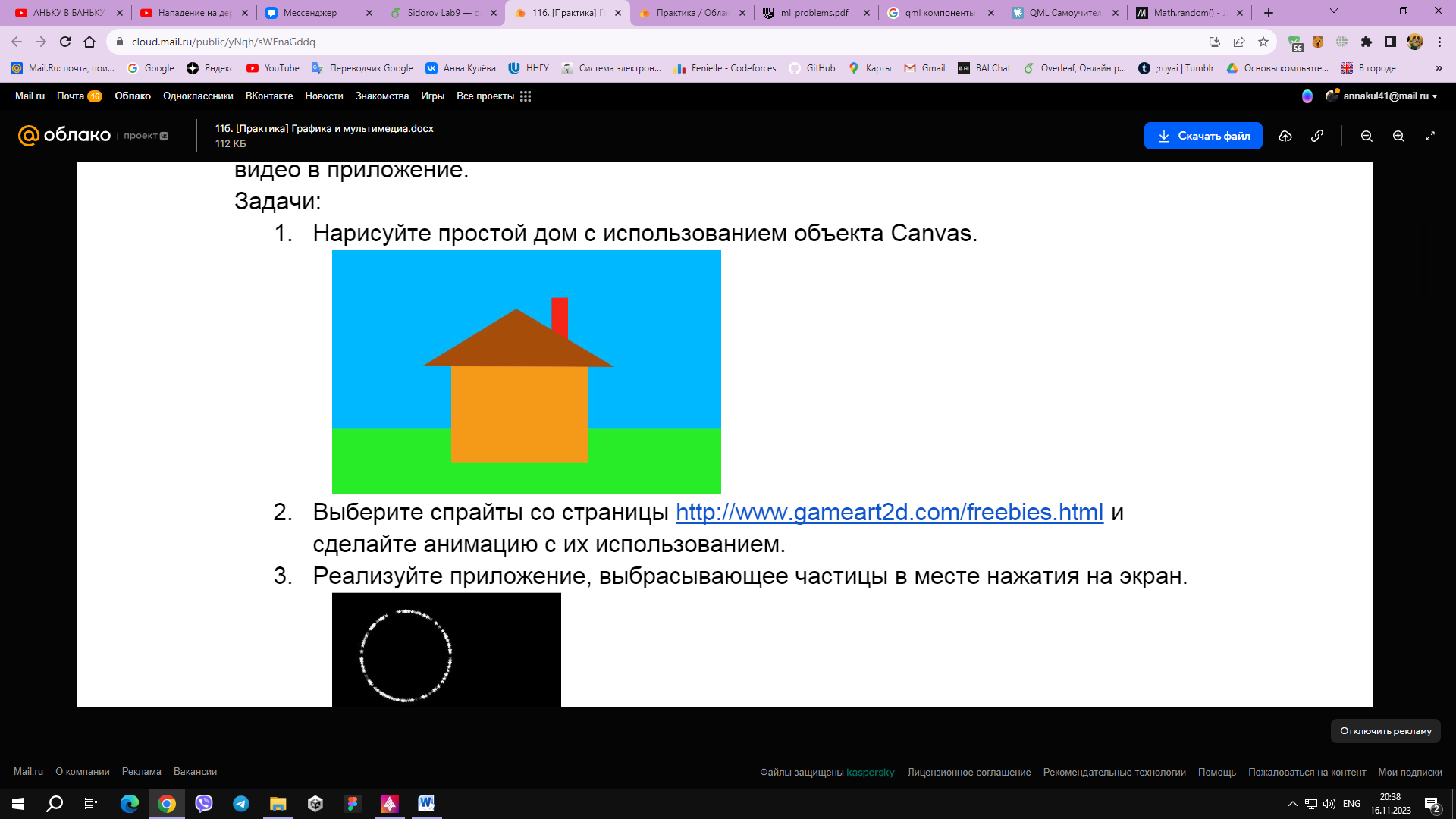
[Приложение 13](#_Toc151220356)

# Цель практического занятия

Цель данного практического занятия состоит в том, чтобы научиться использовать объект Canvas для рисования. Освоить реализацию анимации с помощью спрайтов. Научиться использовать частицы для создания визуальных эффектов. Изучить объекты мультимедиа для встраивания музыки и видео в приложение.

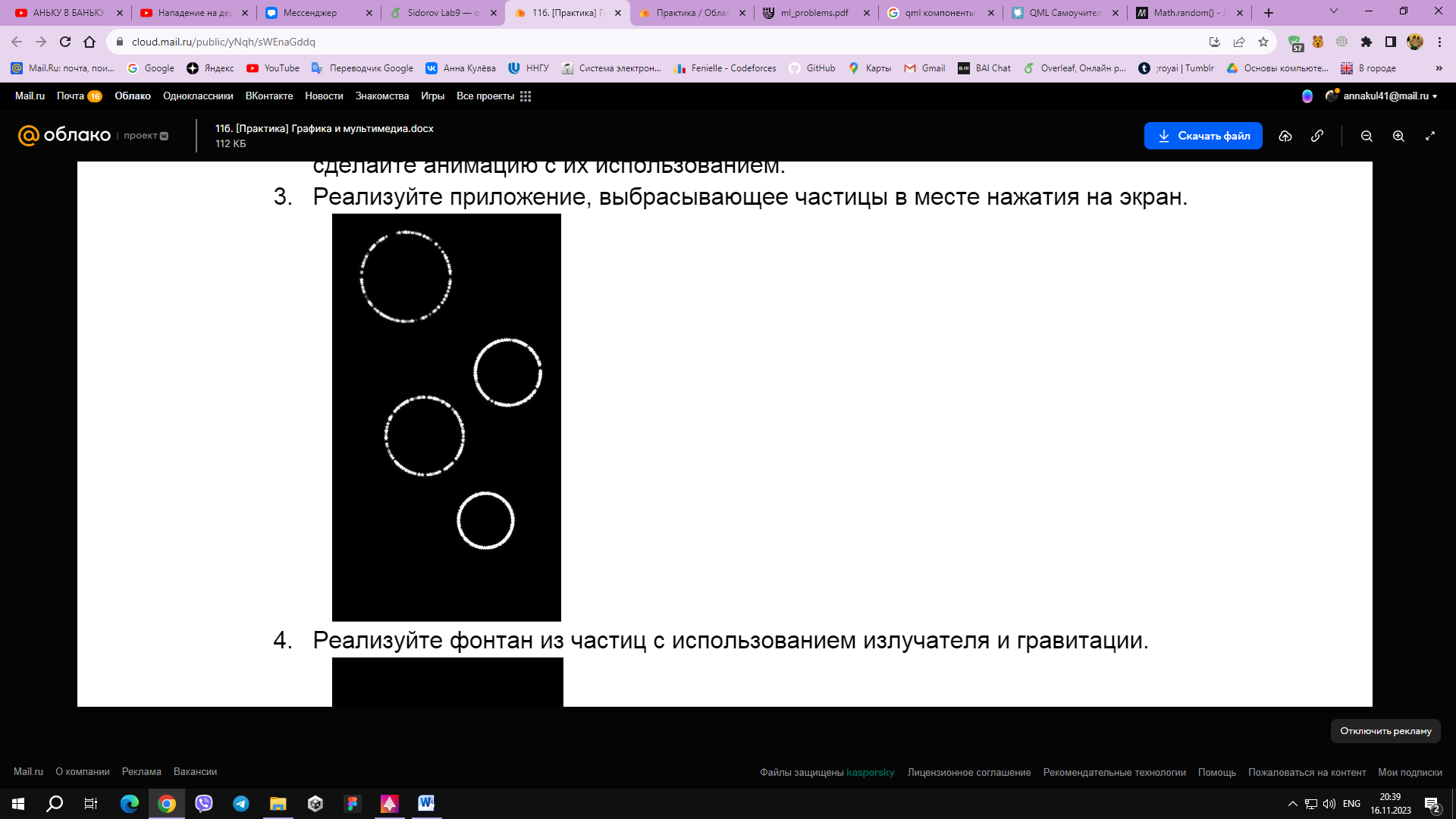
# Постановка задачи

1. Нарисуйте простой дом с использованием объекта Canvas.

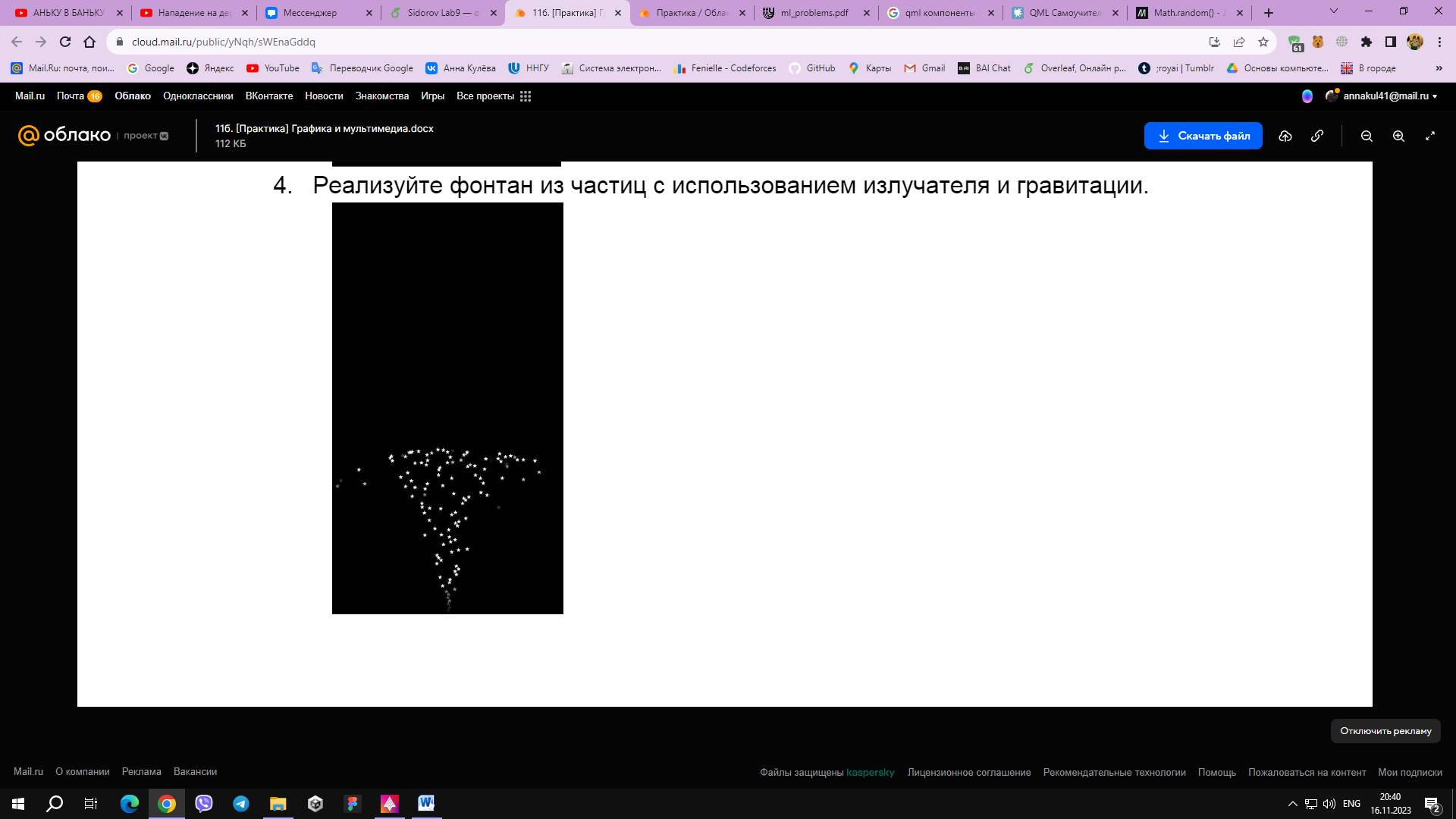


2. Выберите спрайты со страницы http://www.gameart2d.com/freebies.html и сделайте анимацию с их использованием.

3. Реализуйте приложение, выбрасывающее частицы в месте нажатия на экран.



4. Реализуйте фонтан из частиц с использованием излучателя и гравитации.



5. Реализуйте плеер для проигрывания музыкального файла. Музыкальные файлы можно найти на странице http://www.stephaniequinn.com/samples.htm.

6. Реализуйте приложение для воспроизведения видео с помощью объекта Video. Видео доступно по ссылке http://clips.vorwaerts-gmbh.de/big\_buck\_bunny.ogv.

7. Для приложения из задания No5 добавьте возможность переключения между музыкальными файлами для проигрывания с помощью объекта PlayList.

# Руководство пользователя

При запуске программы пользователь увидит главную страницу с кнопками, по которым сможет перейти к 1, 2, 3 и 4 заданиям. На странице с заданием 1 будет отображаться рисунок, нарисованный с помощью объекта Canvas (рисунок 1).

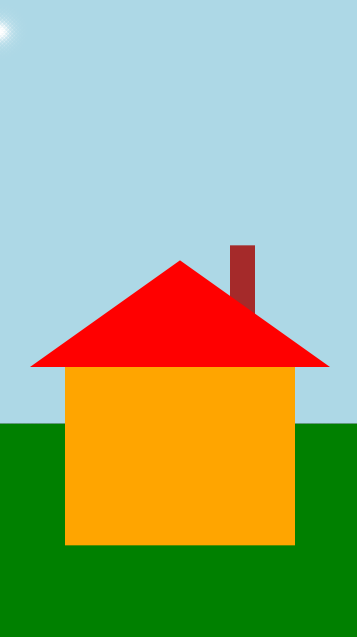


Рисунок 1. Дом Canvas

На странице с заданием 2 будет отображаться анимация с использованием спрайтов (рисунок 2).



Рисунок 2. Анимация спрайтов

На странице с заданием 3 по клику будут появляться частицы в форме окружности (рисунок 3).



Рисунок 3. Выброс частиц по клику

На странице с заданием 4 отображается фонтан из частиц (рисунок 4).



Рисунок 4. Фонтан частиц

# Руководство программиста

Программа реализована на языке программирования QML.

1. Реализуем код, который создаёт страницу с элементом Canvas, который рисует различные фигуры и закрашивает их определенными цветами.

- Изначально указывается размер холста, который равен ширине и высоте родителя.

- Далее объявляется обработчик onPaint, который запускается при перерисовке холста.

- Внутри обработчика создается контекст рисования 2D.

- С помощью fillRect рисуется синий прямоугольник верхней части холста.

- Затем рисуется зеленый прямоугольник нижней части холста.

- Следующий прямоугольник оранжевый и находится в центре холста.

- Последние два прямоугольника - коричневый и красный - рисуются с использованием метода fillRect и метода beginPath для создания пути линий.

- В пути линий, используется метод moveTo для указания начальной точки, и методы lineTo для отрисовки линий. После чего путь замыкается с помощью метода closePath и заполняется красным цветом.

1. Для создания анимации используются спрайты Icon1-5 (рисунок 5).

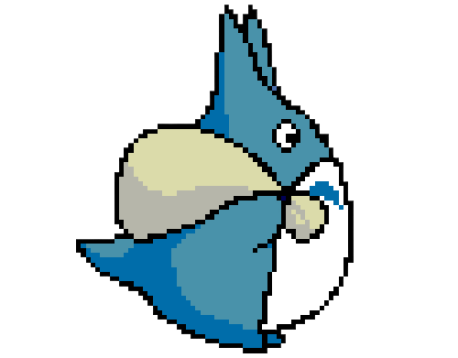
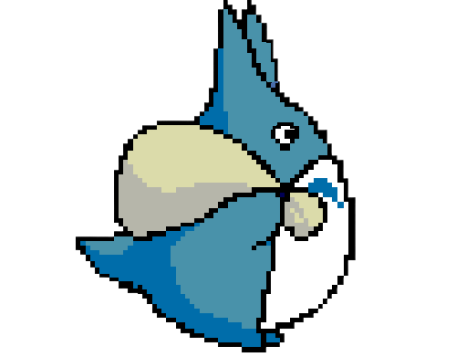
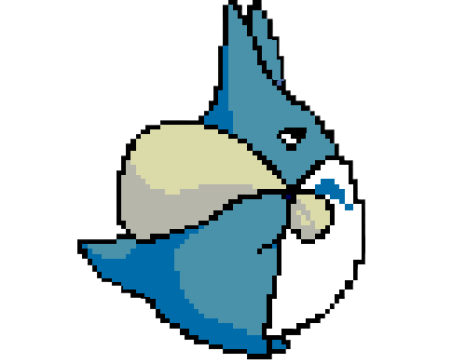
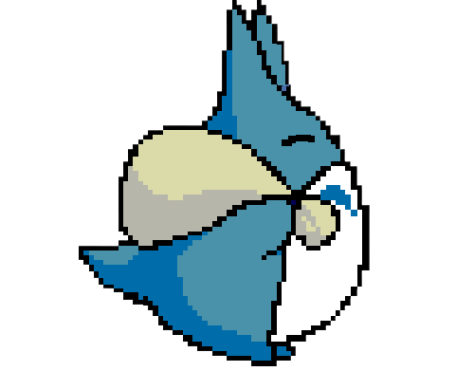
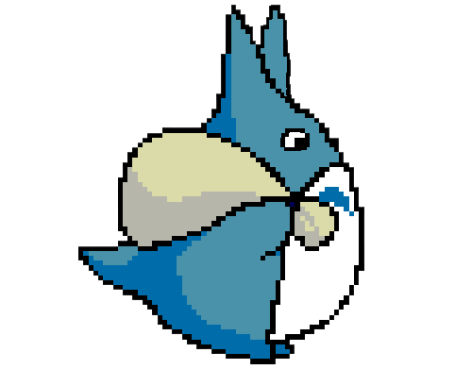


Рисунок 5. Спрайты для анимации

В последовательной анимации SequentialAnimation каждый спрайт показывается на 1/5 долю секунды. Затем с помощью ScriptAction полю source объекта Image присваивается следующий спрайт. Так последовательно меняются 5 изображений, сама же анимация зациклена с помощью loops: Animation.Infinite.

1. Для создания выброса частиц по нажатию мышки выполняем следующие шаги:

- Внутри компонента "particles" создается контейнер "container" размером 200x200 пикселей.

- Внутри контейнера создается система частиц (ParticleSystem), которая будет управлять отображением и движением частиц.

- Далее создается эмиттер (Emitter), который определяет параметры генерации частиц: ширина и высота области, из которой они будут генерироваться, скорость, длительность жизни и т.д.

- Следующий элемент - ItemParticle - определяет визуальное представление каждой частицы, в данном случае - прямоугольник желтого цвета.

- Затем добавляется таймер, который через каждую секунду уничтожает контейнер, чтобы остановить генерацию частиц.

- На следующем уровне вложенности определяется область, в которой можно кликнуть мышью. При каждом клике создается новый объект ParticleSystem, который отображает анимацию частиц, и он размещается на месте клика.

1. Для создания фонтана частиц выполняем следующие шаги:

- Сначала объявляется объект ParticleSystem с идентификатором "particleSystem".

- Затем создается Emitter, который представляет источник частиц. Его размеры и расположение задаются с помощью свойств width, height, anchors.bottom и anchors.horizontalCenter. Частицы, созданные источником, будут существовать в течение определенного времени, которое указывается в свойстве lifeSpan (в миллисекундах) и будет варьироваться в рамках указанного диапазона (lifeSpanVariation). Также устанавливается скорость частиц, которая определяется объектом AngleDirection с заданным углом, изменчивостью угла (angleVariation) и величиной скорости (magnitude).

- Для создания эффекта притяжения курсора мыши указан Gravity, который применяется ко всей системе частиц. Магнитуда (magnitude) и угол (angle) устанавливаются для определения силы притяжения.

- Отображение каждой отдельной частицы определяется с помощью ItemParticle. Внутри ItemParticle используется простой прямоугольник (Rectangle) с цветом "dodgerblue", размером и радиусом, которые зависят от свойства particleSize, устанавливаемого внутри делегата.

# Заключение

В данной лабораторной работе я научилась использовать объект Canvas для рисования. Освоить реализацию анимации с помощью спрайтов. Научиться использовать частицы для создания визуальных эффектов. Изучить объекты мультимедиа для встраивания музыки и видео в приложение. Также были выполнены все шаги практического задания.

# Приложение

**MainPage.qml**

*import* QtQuick 2.0

*import* Sailfish.Silica 1.0

Page {

Column {

anchors.centerIn: *parent*;

spacing: 20;

Button {

text: "Задание 1"

anchors.horizontalCenter: *parent*.horizontalCenter;

onClicked: pageStack.push(*Qt*.resolvedUrl(*qsTr*("CanvasHouse.qml")))

}

Button {

text: "Задание 2"

anchors.horizontalCenter: *parent*.horizontalCenter;

onClicked: pageStack.push(*Qt*.resolvedUrl(*qsTr*("Sprite.qml")))

}

Button {

text: "Задание 3"

anchors.horizontalCenter: *parent*.horizontalCenter;

onClicked: pageStack.push(*Qt*.resolvedUrl(*qsTr*("Tap.qml")))

}

Button {

text: "Задание 4"

anchors.horizontalCenter: *parent*.horizontalCenter;

onClicked: pageStack.push(*Qt*.resolvedUrl(*qsTr*("Fountain.qml")))

}

}

}

**CanvasHouse.qml**

*import* QtQuick 2.0

*import* Sailfish.Silica 1.0

Page {

Canvas {

width: *parent*.width

height: *parent*.height

onPaint: {

*var* *ctx* = *getContext*("2d");

*ctx*.fillStyle = "lightblue";

*ctx*.fillRect(0, 0, *width*, *height*/1.5);

*ctx*.fillStyle = "green"

*ctx*.fillRect(0, *height*/1.5, *width*, *height*-*height*/1.5)

*ctx*.fillStyle = "orange";

*ctx*.fillRect(*width*/2-230, *height*/3+310, 460, 360)

*ctx*.fillStyle = 'brown'

*ctx*.fillRect(*width*/2+100, *height*/3+70, 50, 150)

*ctx*.fillStyle = 'red'

*ctx*.beginPath()

*ctx*.moveTo(*width*/2, *height*/3+100);

*ctx*.lineTo(*width*/2+300, *height*/2+100);

*ctx*.lineTo(*width*/2-300, *height*/2+100);

*ctx*.fill();

*ctx*.closePath();

}

}

}

**Sprite.qml**

*import* QtQuick 2.0

*import* Sailfish.Silica 1.0

Page {

Image {

id: *img1*

anchors.centerIn: *parent*

source: "Idol1.png"

fillMode: Image.PreserveAspectFit

}

SequentialAnimation {

running: true

loops: Animation.Infinite

NumberAnimation {

target: *img1*

property: "x"

from: 0

to: 0

duration: 1000/5

easing.type: Easing.InOutQuad

}

ScriptAction {

script: {

*img1*.source = "Idol2.png"

}

}

NumberAnimation {

target: *img1*

property: "x"

from: 0

to: 0

duration: 1000/4

easing.type: Easing.InOutQuad

}

ScriptAction {

script: {

*img1*.source = "Idol3.png"

}

}

NumberAnimation {

target: *img1*

property: "x"

from: 0

to: 0

duration: 1000/4

easing.type: Easing.InOutQuad

}

ScriptAction {

script: {

*img1*.source = "Idol4.png"

}

}

NumberAnimation {

target: *img1*

property: "x"

from: 0

to: 0

duration: 1000/4

easing.type: Easing.InOutQuad

}

ScriptAction {

script: {

*img1*.source = "Idol5.png"

}

}

NumberAnimation {

target: *img1*

property: "x"

from: 0

to: 0

duration: 1000/4

easing.type: Easing.InOutQuad

}

ScriptAction {

script: {

*img1*.source = "Idol1.png"

}

}

}

}

**Tap.qml**

*import* QtQuick 2.0

*import* Sailfish.Silica 1.0

*import* QtQuick.Particles 2.0

Page {

Component {

id: *particles*

Item {

id: *container*

width: 200

height: 200

ParticleSystem { id: *particleSystem* }

Emitter {

system: *particleSystem*

width: 10

height: 10

emitRate: 2000

maximumEmitted: 200

lifeSpan: 2000

velocity: TargetDirection {

targetX: 0

targetY: 0

targetVariation: 360

magnitude: 100

}

}

ItemParticle {

system: *particleSystem*

delegate: Rectangle {

*property* *int* particleSize: 5

width: *particleSize*

height: *particleSize*

color: "lightyellow"

radius: *particleSize*/2

}

}

Timer {

interval: 1000

running: true

onTriggered: *container*.destroy();

}

}

}

MouseArea {

anchors.fill: *parent*

onClicked: {

*const* *object* = *particles*.createObject(*parent*);

*object*.x = *mouseX*

*object*.y = *mouseY*

}

}

}

**Fountain.qml**

*import* QtQuick 2.0

*import* Sailfish.Silica 1.0

*import* QtQuick.Particles 2.0

Page {

ParticleSystem {

id: *particleSystem*

}

Emitter {

system: *particleSystem*

width: 10

height: 10

anchors.bottom: *parent*.bottom

anchors.horizontalCenter: *parent*.horizontalCenter

lifeSpan: 9000

lifeSpanVariation: 1000

emitRate: 50

velocity: AngleDirection {

angle: -90

angleVariation: 10

magnitude: 300

}

}

Gravity {

system: *particleSystem*

anchors.fill: *parent*

angle: 90

magnitude: 50

}

ItemParticle {

system: *particleSystem*

delegate: Rectangle {

*property* *int* particleSize: 8

width: *particleSize*

height: *particleSize*

color: "dodgerblue"

radius: *particleSize*/2

}

}

}