МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики. Кафедра програмного забезпечення комп'ютерних систем

# ЗВІТ
## з лабораторної роботи № 6
## «СТВОРЕННЯ АНІМАЦІЙ У ЗАСТОСУНКАХ ANDROID»

**Виконав:**
студент 3-го курсу, групи КП-83,
спеціальності 121 – Інженерія
програмного забезпечення
Коваль Андрій Олександрович

**Перевірив:**
к. т. н, старший викладач
Хайдуров Владислав
Володимирович

Київ – 2020

# ВСТУП

Метою даної роботи є ознайомлення із основними принципами створення анімацій у програмних застосунках під ОС Android та навчитись створювати анімації власних графічних елементів дизайну. Набуті навички дозволять розширити знання про можливості малювання в Android та поглиблять знання алгоритмів малювання й відображення об’єктів.

## Завдання до лабораторної роботи

1. Ознайомитись із усіма теоретичними відомостями до лабораторної роботи.
2. Переглянути усі практичні завдання при роботі з графічними об’єктами та методами їх створення.
3. Створити проекти з усіма розглянутими у лабораторній роботі анімаціями, продемонструвати їх роботу. Виконати відповідні скріншоти та додати їх до звіту лабораторної роботи.
4. Створити мобільний застосунок, у якому квадрат рухається (заданий довжиною сторони у пікселях у програмі) по колу (параметри кола задаються розробником у програмі), причому центр квадрата лежить на самому колі і квадрат не обертається відносно свого центра. Кольори обрати на власний розсуд. У додатку передбачити зміну швидкості (кроку пересування) квадрата по колу.
5. Створити програмний додаток, відтворює «килим» Серпинського: http://hpc-education.ru/files/lectures/2011/ershov/ershov_2011_slides02.pdf. Відтворення «килиму» можна виконувати як з анімацією, так і без неї (за бажанням розробника). За додаткові бали можна реалізувати й інші (по 1 балу за кожен) клітинні автомати з даного джерела.
6. За додаткові бали (до 4 балів) створити гру «Змійка», у якій елементами робочого поля є прозорі прямокутники (з границею певного кольору). Управління «змійкою» можна виконувати за допомогою чотирьох кнопок (типу «Джойстик»). Також передбачити зміну швидкості з часом або за бажанням користувача застосунку. Для бажаючих отримати ще додаткові бали (до 4 балів), необхідно оформити даний програмний додаток окремо у вигляді мініпроекту:

короткий звіт (за стилем лабораторної роботи) та створити презентацію, що містить 5-10 слайдів. Дедлайн актуальності виконання мініпроекту – тиждень після дедлайну здачі звітів даної лабораторної роботи до classroom.

**Завдання за додаткові бали**

7. За додаткові бали (до 4-х) програмно реалізувати наступне завдання.

   Поширення інфекції. Нехай є ділянка шкіри розміром n × n (n – непарне) клітин. Передбачається, що вихідною зараженою кліткою шкіри є центральна. У кожен інтервал часу уражена інфекцією клітина може з ймовірністю 0,5 заразити будь-яку з сусідніх здорових клітин. Після шести одиниць часу заражена клітина стає несприйнятливою до інфекції. Виниклий імунітет діє протягом наступних чотирьох одиниць часу, а потім клітина виявляється здоровою. В ході моделювання описаног процесу видавати поточний стан модельованого ділянки шкіри в кожному інтервалі
   часу відзначаючи заражені, несприйнятливі до інфекції і здоров клітини (кольором клітинки).

   Вивести динаміку поширення інфекції з використанням елементу
   Rectangle та Animation у Canvas.

   Для бажаючих отримати ще додаткові бали (до 4 балів), необхідно оформити даний програмний додаток окремо у вигляді мініпроекту: короткий звіт (за стилем лабораторної роботи) та створити презентацію, що містить 5-10 слайдів. Дедлайн актуальності виконання мініпроекту – тиждень після дедлайну здачі звітів даної лабораторної роботи до classroom.

8. За додаткові бали (до 4-х) програмно реалізувати наступне завдання.

Сегрегація населення. Припустимо, що досліджуваний регіон може бути представлений у сигляді решітки m × n, де кожна клітина відповідає одному будинку. Припустимо також, що кожен будинок може бути зайнятий білою (o) або чорною (x) сім'єю, або залишитися порожнім. У даній моделі є три можливих стану. У даному завданні вважається, що кожна расова група хоче мати не менше вказаної кількості відсотків сусідів з тим же кольором шкіри. Якщо ця умова не виконалась, то сім'я перебирається в найближчий дім, де відсотковий склад сусідів є прийнятним. Вважається, що розумний вибір можна зробити, якщо в данному поселенні 25%–30 будинків не заселені. Розглянути два правила поведінки жителів, які оцінюють відсоток прийнятних сусідів (використовути окіл Мура кількість сусідніх клітинок рівна 8 для поточної клітинки):

1) кількість сусідів будинків, заселених представниками однії і тієї ж раси, повинна бути не менша третини;
2) кількість сусідів будинків, заселених представниками однії і тієї ж раси, повинна бути не менша половини.

Вивести динаміку переселення з використанням елементу Rectangle та Animation у Canvas.

Для бажаючих отримати ще додаткові бали (до 4 балів), необхідно оформити даний програмний додаток окремо у вигляді мініпроекту: короткий звіт (за стилем лабораторної роботи) та створити презентацію, що містить 5-10 слайдів. Дедлайн актуальності виконання мініпроекту – тиждень після дедлайну здачі звітів даної лабораторної роботи до classroom.

## Короткі теоретичні відомості

Основним методом малювання у роботі було створення власного класу який наслідується від класу View та реалізує необхіжний метод onDraw, якому у параметры передається об’єкт типу Canvas, на якому мы можемо виконувати усі необхідні нам маніпуляції за допомогою об’єкта Paint, що являє собою аналог пензлика для малювання та методу invalidate() у View, що змушує UI перемальовуватись.

## Програмна реалізація задачі

Було реалізовано три завдання:

- Основні анімації у вигляді квадратів, що рухаються, відбиваються від стінок та динамічно змінюють свій розмір;
- Обертання квадрату по колу
- Генерація та малювання клітинних автоматів з будь-якими вхідними параметрами. Було реалізовано універсальний алгоритм побудови та відображення автоматів в залежності від вхідних умов таких як розміри пікселів, полотна та умов, за якими обираються наступні рівні.

### Завдання 1

У цьому завдання випадковим чином створюються деяка кількість квадратів, які мають випадкову швидкість та напрям. Вони рухаться, відбиваються від стін та з випадковою швидкістю змінюють свої розміри. Таку поведінку було досягнуто завдяки власному класу MyRectangle

MyRectangle

```
public class MyRectangle {
    private float x;
    private float y;

    private float width;
    private float height;

    private float maxWidth;
    private float maxHeight;

    private float vx;
    private float vy;

    private float dwidth;
    private float dheight;

   //…


    public MyRectangle(float x, float y, float width, float height, float vx, float vy,
float dwidth, float dheight) {
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
```

```
        this.maxWidth = width;
        this.maxHeight = height;
        this.vx = vx;
        this.vy = vy;
        this.dwidth = dwidth;
        this.dheight = dheight;
    }

    public MyRectangle(float x, float y, float width, float height) {
        this(x, y, width, height, 0, 0, 0, 0);
    }

    public void draw(Canvas c, Paint p) {
        c.drawRect(x, y, x + width, y - height, p);
    }

    public void update(Canvas c) {
        if (this.width >= this.maxWidth || this.width <= 0) {
            this.dwidth = -this.dwidth;
        }
        if (this.height >= this.maxHeight || this.height <= 0) {
            this.dheight = -this.dheight;
        }

        if (x + width >= c.getWidth() || x <= 0) {
            this.vx = -this.vx;
        }
        if (y >= c.getHeight() || y - height <= 0) {
            this.vy = -this.vy;
        }

        this.x += this.vx;
        this.y += this.vy;

        this.width += this.dwidth;
        this.height += this.dheight;
    }
}
```

На кожній ітерації квадрати оновлюються та перемальовуються

Процес малювання

```
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    if (rects == null) {
        rects = createRectangles(canvas.getWidth(), canvas.getHeight());
    }
    Paint paint = new Paint();
    paint.setColor(Color.RED);
    paint.setStyle(Paint.Style.STROKE);
    paint.setStrokeWidth(5);
    for (MyRectangle rect : rects) {
        rect.draw(canvas, paint);
    }
    for (MyRectangle rect : rects) {
        rect.update(canvas);
    }
    invalidate();
```

```
}
```

Приклад результату

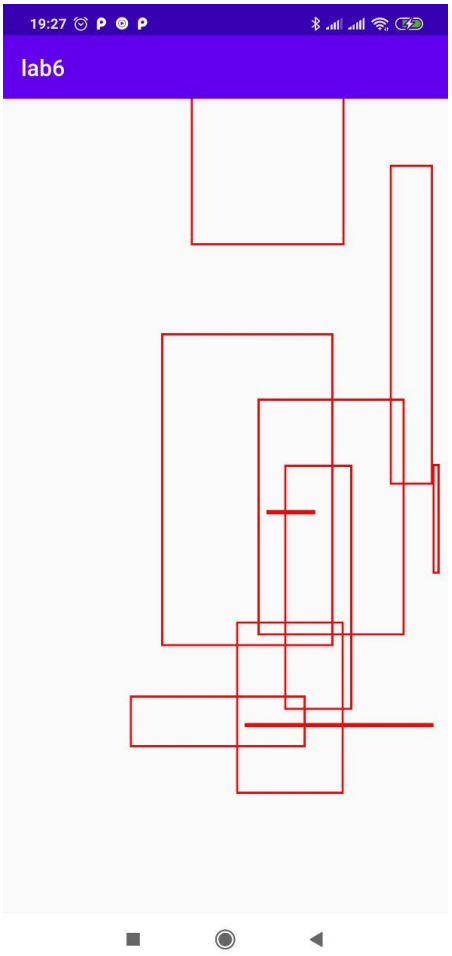


Рис 1-1. Рух квадратів

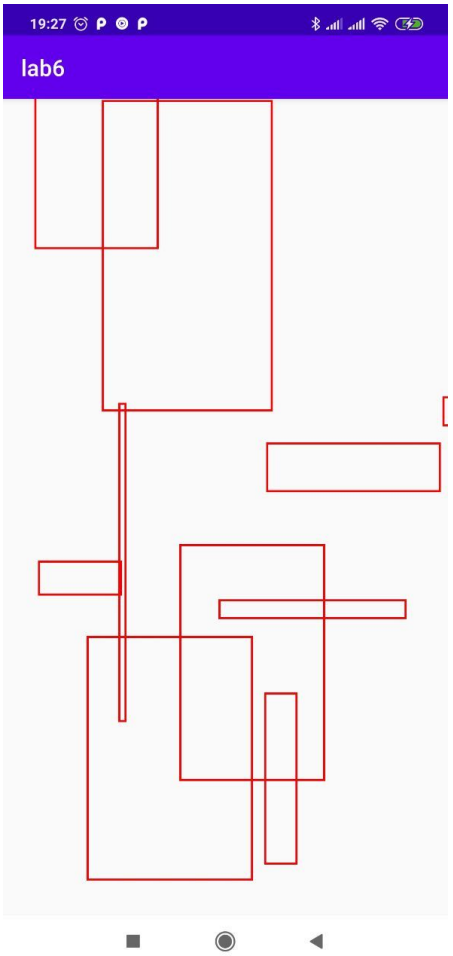


Рис 1-2. Рух квадратів

## Завдання 2

У цьому завданні квадрат, як і очікувалося б, рухається завдяки зміні певного кута, який змінюється за певною кутовою швидкістю.

```
protected void onDraw(Canvas canvas) {
    if (this.rect == null) {
        this.rect = createRect(canvas);
        cx = getWidth() / 2.0;
```

```
        cy = getHeight() / 2.0;
    }
    rectCX = radius * Math.cos(angle);
    rectCY = radius * Math.sin(angle);
    angle += speed;
    if (angle > Math.PI * 2) {
        angle = 0;
    }
    rect.setCX((float)cx + (float)rectCX);
    rect.setCY((float)cy + (float)rectCY);
    paint.setStrokeWidth(10);
    rect.draw(canvas, paint);
    paint.setStrokeWidth(20);
    canvas.drawPoint((float)cx, (float)cy, paint);
    invalidate();
}
```

Змінні angle та speed винесені окремо, та можут бути модіфікованими лише з коду. Нажаль не вийшло додати SeekBar для зміни швидкості, тобто розмістити наш View на розмітці. Резльльтат роботи побачимо на рис 2-(1-2)

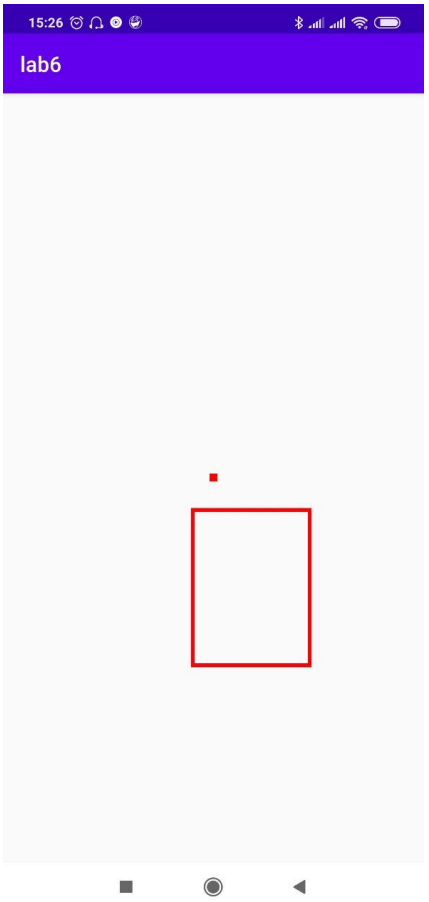


Рис 2-1. Обертання квадрату

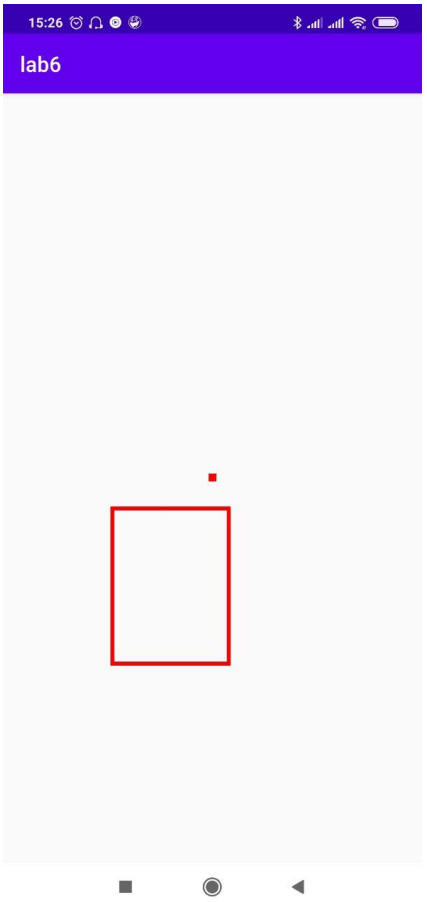


Рис 2-2. Обертання квадрату

## Завдання 3

У цьому завданні було реалізовано окремий клас CellularAutomata для роботи з автоматом. Це було зроблено для спрощення та збільшення рівня абстракції.

CellularAutomata

```
public class CellularAutomata {

    private float left, top, right, bottom, cellSize;

    private int step;

    private int[][] cells;

    private ConditionCell[] conditionCells;

    public CellularAutomata(float left, float right, float top, float bottom, float
cellSize, ConditionCell[] conditionCells) {
        this.left = left;
        this.right = right;
        this.top = top;
        this.bottom = bottom;
        this.cellSize = cellSize;
        this.step = 0;
        this.conditionCells = conditionCells;
        this.initCells(cellSize);
    }

    public void draw(Canvas canvas, Paint paint) {
        float prevStrokeWidth = paint.getStrokeWidth();
        Paint.Style prevFillStyle = paint.getStyle();

        paint.setStrokeWidth(1);
        paint.setStyle(Paint.Style.FILL);

        for (int i = 0; i < this.cells.length; i++) {
            for (int j = 0; j < this.cells[i].length; j++) {
                float left = j * cellSize + this.left;
                float top = i * cellSize + this.top;
                if (this.cells[i][j] == 1 && left + cellSize <= this.right && top +
cellSize <= this.bottom) {
                    canvas.drawRect(left, top, left + cellSize, top + cellSize, paint);
                }
            }
        }

        paint.setStrokeWidth(prevStrokeWidth);
        paint.setStyle(prevFillStyle);
    }

    public boolean update() {
        if (this.step + 1 >= this.cells.length) {
            return false;
        }

        int conditionLength = this.conditionCells[0].condition.length;
```

```
        for (int i = 0; i < this.cells[this.step].length - conditionLength; i++) {
            int[] condition = new int[conditionLength];
            for (int j = i; j < conditionLength + i; j++) {
                condition[j - i] = this.cells[this.step][j];
            }
            for (ConditionCell conditionCell : this.conditionCells) {
                if (conditionCell.conditionMatches(condition)) {
                    this.cells[this.step + 1][i + (int)Math.floor(conditionLength /
2.0)] = conditionCell.cell;
                }
            }
        }

        this.step += 1;
        return true;
    }

    public void reset() {
        this.step = 0;
        for (int i = 0; i < this.cells.length; i++) {
            for (int j = 0; j < this.cells[i].length; j++) {
                this.cells[i][j] = 0;
            }
        }
        this.cells[0][this.cells[0].length / 2] = 1;
    }

    private void initCells(float cellSize) {
        float width = this.right - this.left;
        float height = this.bottom - this.top;

        int cellsWidth = (int)Math.floor(width / cellSize);
        int cellsHeight = (int)Math.floor(height / cellSize);

        this.cells = new int[cellsHeight][cellsWidth];
        this.cells[0][cellsWidth / 2] = 1;
    }

    public static class ConditionCell {

        int[] condition;
        int cell;

        public ConditionCell(int[] condition, int cell) {
            this.condition = condition;
            this.cell = cell;
        }

        public boolean conditionMatches(int[] cells) {
            for (int i = 0; i < this.condition.length; i++) {
                if (cells[i] != this.condition[i]) {
                    return false;
                }
            }
            return true;
        }
    }
}
```

В результаті чого, код для малювання та оновлення автоматів є простим та лаконічним

Малювання та оновлення автоматів

```
@Override
protected void onDraw(Canvas canvas) {
    if (this.automatas == null) {
        this.automatas = createAutomatas(getWidth(), getHeight());
    }
    for (CellularAutomata automata : this.automatas) {
        automata.draw(canvas, paint);
        if (!automata.update()) {
            automata.reset();
        }
    }
    invalidate();
}
```

Приклад створення автомату

```
new CellularAutomata(
             0,
             width / 2,
             0,
             height / 3,
             5,
             new CellularAutomata.ConditionCell[] {
               new CellularAutomata.ConditionCell(new int[] {1, 1, 1}, 0),
               new CellularAutomata.ConditionCell(new int[] {1, 1, 0}, 1),
               new CellularAutomata.ConditionCell(new int[] {1, 0, 1}, 0),
               new CellularAutomata.ConditionCell(new int[] {1, 0, 0}, 1),
               new CellularAutomata.ConditionCell(new int[] {0, 1, 1}, 1),
               new CellularAutomata.ConditionCell(new int[] {0, 1, 0}, 0),
               new CellularAutomata.ConditionCell(new int[] {0, 0, 1}, 1),
               new CellularAutomata.ConditionCell(new int[] {0, 0, 0}, 0)
          }),
```

Так ми створюємо 6 автоматів, даємо їм їх координати на полотні та спостерігаємо за вимальвуванням (рис 3-(1-2))

```
private CellularAutomata[] createAutomatas(float width, float height) {
    return new CellularAutomata[] {
            new CellularAutomata(
                    0,
                    width / 2,
                    0,
                    height / 3,
```

```
                5,
                new CellularAutomata.ConditionCell[] {
                        new CellularAutomata.ConditionCell(new int[] {1, 1, 1}, 0),
                        new CellularAutomata.ConditionCell(new int[] {1, 1, 0}, 1),
                        new CellularAutomata.ConditionCell(new int[] {1, 0, 1}, 0),
                        new CellularAutomata.ConditionCell(new int[] {1, 0, 0}, 1),
                        new CellularAutomata.ConditionCell(new int[] {0, 1, 1}, 1),
                        new CellularAutomata.ConditionCell(new int[] {0, 1, 0}, 0),
                        new CellularAutomata.ConditionCell(new int[] {0, 0, 1}, 1),
                        new CellularAutomata.ConditionCell(new int[] {0, 0, 0}, 0)
                }),
        new CellularAutomata(
                width / 2,
                width,
                0,
                height / 3,
                10,
                new CellularAutomata.ConditionCell[] {
                        new CellularAutomata.ConditionCell(new int[] {1, 1, 1}, 1),
                        new CellularAutomata.ConditionCell(new int[] {1, 1, 0}, 0),
                        new CellularAutomata.ConditionCell(new int[] {1, 0, 1}, 0),
                        new CellularAutomata.ConditionCell(new int[] {1, 0, 0}, 1),
                        new CellularAutomata.ConditionCell(new int[] {0, 1, 1}, 0),
                        new CellularAutomata.ConditionCell(new int[] {0, 1, 0}, 1),
                        new CellularAutomata.ConditionCell(new int[] {0, 0, 1}, 1),
                        new CellularAutomata.ConditionCell(new int[] {0, 0, 0}, 0)
                }),
        new CellularAutomata(
                0,
                width / 2,
                height / 3,
                height / 3 * 2,
                5,
                new CellularAutomata.ConditionCell[] {
                        new CellularAutomata.ConditionCell(new int[] {1, 1, 1}, 0),
                        new CellularAutomata.ConditionCell(new int[] {1, 1, 0}, 0),
                        new CellularAutomata.ConditionCell(new int[] {1, 0, 1}, 0),
                        new CellularAutomata.ConditionCell(new int[] {1, 0, 0}, 1),
                        new CellularAutomata.ConditionCell(new int[] {0, 1, 1}, 1),
                        new CellularAutomata.ConditionCell(new int[] {0, 1, 0}, 1),
                        new CellularAutomata.ConditionCell(new int[] {0, 0, 1}, 1),
                        new CellularAutomata.ConditionCell(new int[] {0, 0, 0}, 0)
                }),
        new CellularAutomata(
                width / 2,
                width,
                height / 3,
                height / 3 * 2,
                4,
                new CellularAutomata.ConditionCell[] {
                        new CellularAutomata.ConditionCell(new int[] {1, 1, 1}, 0),
                        new CellularAutomata.ConditionCell(new int[] {1, 1, 0}, 1),
                        new CellularAutomata.ConditionCell(new int[] {1, 0, 1}, 1),
                        new CellularAutomata.ConditionCell(new int[] {1, 0, 0}, 0),
                        new CellularAutomata.ConditionCell(new int[] {0, 1, 1}, 1),
                        new CellularAutomata.ConditionCell(new int[] {0, 1, 0}, 0),
                        new CellularAutomata.ConditionCell(new int[] {0, 0, 1}, 0),
                        new CellularAutomata.ConditionCell(new int[] {0, 0, 0}, 1)
                }),
        new CellularAutomata(
                0,
                width / 2,
                height / 3 * 2,
                height,
                4,
```

```
                    new CellularAutomata.ConditionCell[] {
                            new CellularAutomata.ConditionCell(new int[] {1, 1, 1}, 0),
                            new CellularAutomata.ConditionCell(new int[] {1, 1, 0}, 1),
                            new CellularAutomata.ConditionCell(new int[] {1, 0, 1}, 1),
                            new CellularAutomata.ConditionCell(new int[] {1, 0, 0}, 0),
                            new CellularAutomata.ConditionCell(new int[] {0, 1, 1}, 1),
                            new CellularAutomata.ConditionCell(new int[] {0, 1, 0}, 1),
                            new CellularAutomata.ConditionCell(new int[] {0, 0, 1}, 1),
                            new CellularAutomata.ConditionCell(new int[] {0, 0, 0}, 0)
                    }),
            new CellularAutomata(
                    width / 2,
                    width,
                    height / 3 * 2,
                    height,
                    3,
                    new CellularAutomata.ConditionCell[] {
                            new CellularAutomata.ConditionCell(new int[] {1, 1, 1}, 0),
                            new CellularAutomata.ConditionCell(new int[] {1, 1, 0}, 0),
                            new CellularAutomata.ConditionCell(new int[] {1, 0, 1}, 1),
                            new CellularAutomata.ConditionCell(new int[] {1, 0, 0}, 1),
                            new CellularAutomata.ConditionCell(new int[] {0, 1, 1}, 1),
                            new CellularAutomata.ConditionCell(new int[] {0, 1, 0}, 0),
                            new CellularAutomata.ConditionCell(new int[] {0, 0, 1}, 0),
                            new CellularAutomata.ConditionCell(new int[] {0, 0, 0}, 1)
                    })
    };
}
```

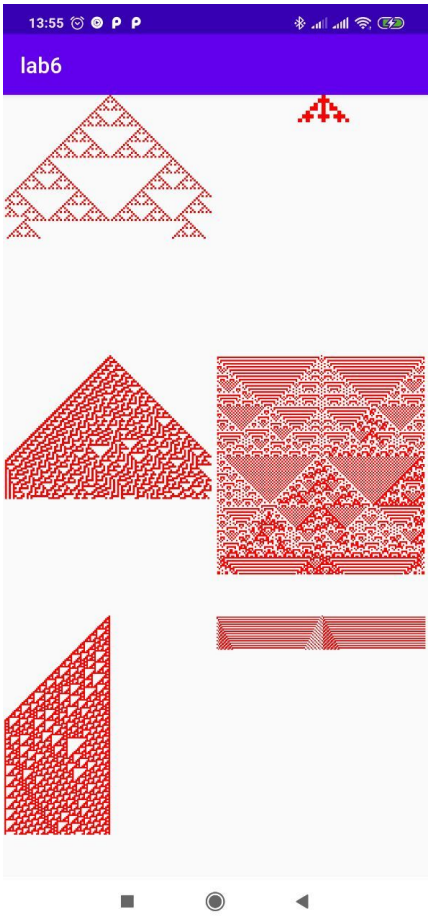


Рис 3-1, малювання клітковихавтоматів

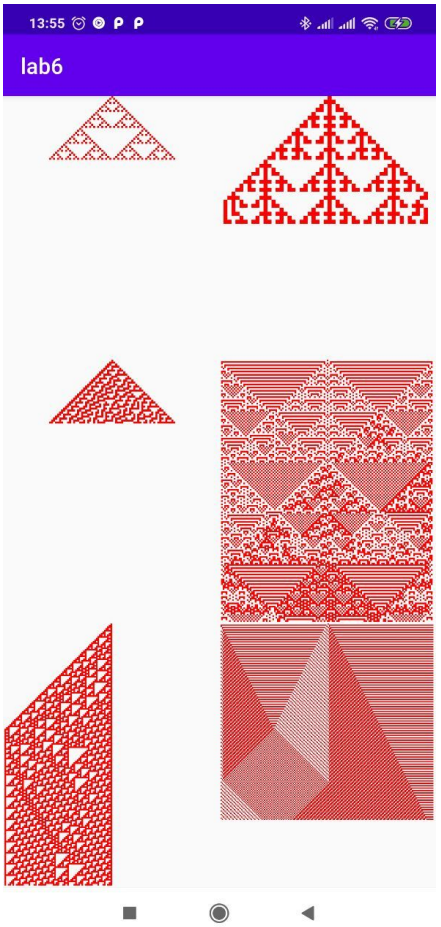


Рис 3-2, малювання клітковихавтоматів

# ВИСНОВКИ

Я ознайомився із основними принципами створення анімацій у програмних застосунках під ОС Android та навчився створювати анімації власних графічних елементів дизайну. Набуті навички дозволять розширити знання про можливості малювання в Android та поглиблять знання алгоритмів малювання й відображення об’єктів.