Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Сибирский государственный университет
телекоммуникаций и информатики»

Лабораторная работа №6

Выполнили: студенты 3 курса
ИВТ, гр. ИП-713
Трусов К.В.
Михеев Н.А.

Новосибирск, 2020 г.

Задание

Необходимо сделать (на выбор) или живые обои или виджет для рабочего стола.

Решение поставленной задачи

Были реализованы живые обои, методом отрисовки на canvas с наличием WallpaperService'а, являющегося частью android API, в котором выполняется рендер обоев в зависимости от статуса устройства.

После установки пользователю предлагается установить обои в настройках.

Сами же обои представляют из себя морской фон и поверх него отрисованы рыбки, которые плавают из стороны в сторону. Для того чтобы получить плавающих, анимированных рыбок был использован спрайт с раскадрированной анимацией плавающей рыбки.

Рис. 1 – пример спрайта-анимации

Далее подробнее по каждому реализованному классу с листингом некоторых важных функций:

- Класс Sea – отвечает за создание рыб, отрисовку заднего плана, рендер самих рыбок, поиск правой и левой границ экрана, дабы рыбки плавали только в границах экрана смартфона.

```
private void addFishes() {
    Point startPoint = new Point( x: 100,  y: 300);
    this._fishes.add(new Fish(this._context,  sea: this, startPoint,  speed: 150));
    Point startPoint1 = new Point( x: 120,  y: 550);
    this._fishes.add(new Fish(this._context,  sea: this, startPoint1,  speed: 250));
    Point startPoint2 = new Point( x: 210,  y: 710);
    this._fishes.add(new Fish(this._context,  sea: this, startPoint2,  speed: 95));
    Point startPoint3 = new Point( x: 310,  y: 950);
    this._fishes.add(new Fish(this._context,  sea: this, startPoint3,  speed: 550));
    Point startPoint4 = new Point( x: 125,  y: 1100);
    this._fishes.add(new Fish(this._context,  sea: this, startPoint4,  speed: 190));
    Point startPoint5 = new Point( x: 160,  y: 1480);
    this._fishes.add(new Fish(this._context,  sea: this, startPoint5,  speed: 330));
}
```

Рис. 2 – метод создания 6 рыбок с заданными координатами старта и скоростью

- Все визуализированные объекты в аквариуме должны реализовывать интерфейс Renderable, который имеет один метод – render().

- Был создан отдельный абстрактный класс SeaAnimal, в котором задавались все необходимые настройки для «морских обитателей» - скорость, направление, инициализация спрайтов для анимации, в нем же происходил рендер полученного спрайта на canvas.
- Сама анимация спрайтов была реализована в отдельном классе – FishSprite.
- Создание рыб (объекта Fish) происходит в классе Sea. Fish – производный класс от SeaAnimal с определенным изображением спрайта. Количество кадров и выбор FPS выделяется в отдельные макросы в нем же.

```
public class Fish extends SeaAnimal {
    private static final int TOTAL_FRAMES_IN_SPRITE = 6;
    private static final int CLOWN_FISH_FPS = 6;
```

Рис. 3 – пример настроек для анимации спрайта

Скриншоты

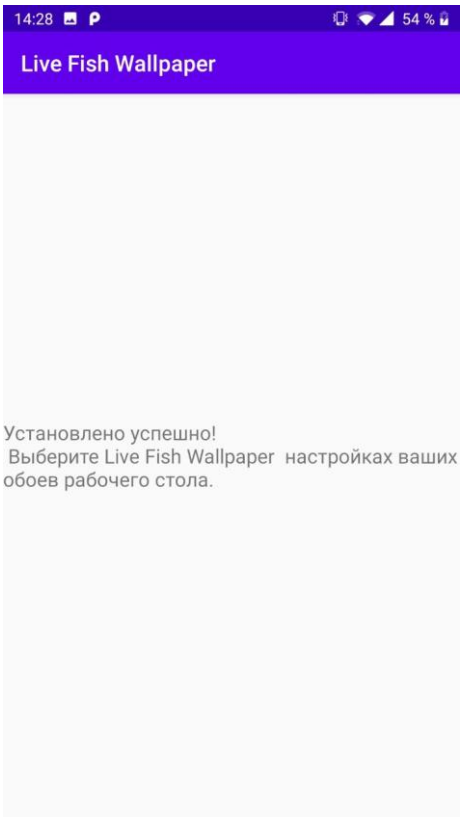

Рис.4 – приветственное сообщение после установки приложения

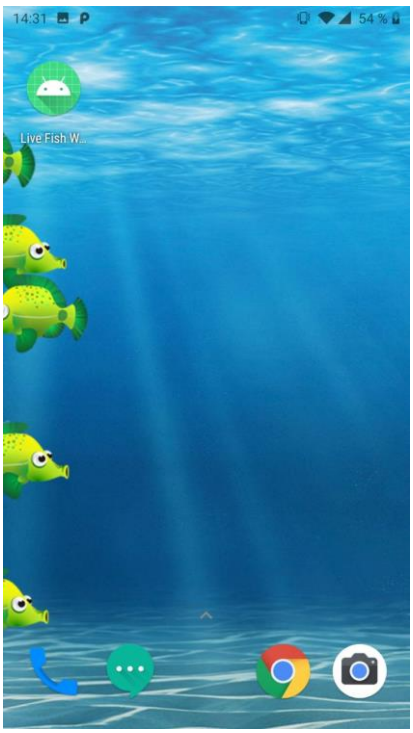

Рис. 5 – пример состояния обоев, когда часть рыбок начала уже плыть в другую сторону, а некоторые еще только заканчивали свой путь

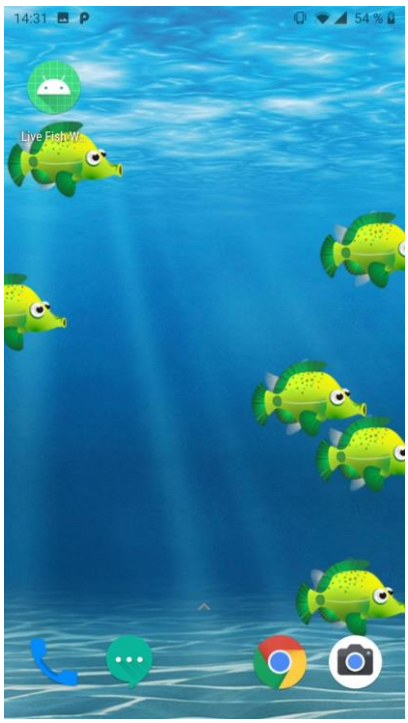

Рис. 6 – пример №2 состояния обоев, рыбки плавают с разной скоростью

Листинг программы

Класс Fish.java:

```
public class Fish extends SeaAnimal {
        private static final int TOTAL_FRAMES_IN_SPRITE = 6;
        private static final int CLOWN_FISH_FPS = 6;


        public Fish(Context context, Sea sea, Point startPoint, int speed){
```

```
                super(context, sea);
                BitmapFactory.Options options = new BitmapFactory.Options();
                options.inPurgeable = true;
                Bitmap     leftBitmap     =     BitmapFactory.decodeResource(getContext().getResources(),
com.sibsutis.walls.R.drawable.left, options);
                BitmapFactory.Options options1 = new BitmapFactory.Options();
                options1.inPurgeable = true;
                Bitmap     rightBitmap     =     BitmapFactory.decodeResource(getContext().getResources(),
com.sibsutis.walls.R.drawable.right, options1);
                this.initialize(leftBitmap,    rightBitmap,    CLOWN_FISH_FPS,    TOTAL_FRAMES_IN_SPRITE,
startPoint, speed);
        }

        public void render(Canvas canvas){
                super.render(canvas);
        }
}
```

Класс FishSprite.java:

```
public class FishSprite {
        private Bitmap _currentSpriteBitmap;
        private Rect _drawRect;
        private int _fps;
        private int _noOfFrames;
        private int _currentFrame;
        private long _timer;
        private int _spriteWidth;
        private int _spriteHeight;
        private Point _position;

        public FishSprite(Bitmap spriteBitmap, int fps, int frameCount, Point startPoint) {

                this.initialize();

                this._position = startPoint;
                this._currentSpriteBitmap = spriteBitmap;
                this._spriteHeight = spriteBitmap.getHeight();
                this._spriteWidth = spriteBitmap.getWidth() / frameCount;
                this._drawRect = new Rect(0,0, this._spriteWidth, this._spriteHeight);
                this._fps = 1000 / fps;
                this._noOfFrames = frameCount;
        }

        private void initialize() {
                this._drawRect = new Rect(0,0,0,0);
                this._timer = 0;
                this._currentFrame = 0;
        }

        private void Update(long currentTime) {
            if(currentTime > this._timer + this._fps ) {
                this._timer = currentTime;
                this._currentFrame +=1;

                if(this._currentFrame >= this._noOfFrames) {
                this._currentFrame = 0;
                }
            }
            this._drawRect.left = this._currentFrame * this._spriteWidth;
            this._drawRect.right = this._drawRect.left + this._spriteWidth;
        }

        public void render(Canvas canvas, long currentTime) {

                this.Update(currentTime);
            Rect dest = new Rect(getXPos(), getYPos(), getXPos() + this._spriteWidth,
                            getYPos() + this._spriteHeight);
            canvas.drawBitmap(this._currentSpriteBitmap, this._drawRect, dest, null);
        }

        public Point getPosition() {
                return _position;
```

```
        }

        public void setPosition(Point position) {
                this._position = position;
        }

        public int getYPos() {
                return this._position.y;
        }

        public int getXPos() {
                return this._position.x;
        }

        public void setYPos(int y) {
                this._position.y = y;
        }

        public void setXPos(int x) {
                this._position.x = x;
        }

        public int getWidth(){
                return this._spriteWidth;
        }

        public int getHeight(){
                return this._spriteHeight;
        }

}
```

Класс SeaWallpaperService.java:

```
public class SeaWallpaperService extends WallpaperService {

   @Override
   public Engine onCreateEngine() {
      return new AquariumWallpaperEngine();
   }

   class AquariumWallpaperEngine extends Engine{

      private Sea _sea;

      public AquariumWallpaperEngine() {
         this._sea = new Sea();
         this._sea.initialize(getBaseContext(), getSurfaceHolder());
      }

      @Override
      public void onVisibilityChanged(boolean visible) {
         if(visible){
            this._sea.render();
         }
      }

      @Override
      public void onSurfaceChanged(SurfaceHolder holder, int format,
            int width, int height) {
         super.onSurfaceChanged(holder, format, width, height);
      }

      @Override
      public void onSurfaceCreated(SurfaceHolder holder) {
         super.onSurfaceCreated(holder);
         this._sea.start();
      }

      @Override
      public void onSurfaceDestroyed(SurfaceHolder holder) {
         super.onSurfaceDestroyed(holder);
         this._sea.stop();
      }
```

```
   }
}
```
+

Класс SeaAnimal.java:

```
public abstract class SeaAnimal implements Renderable {

   private static int MAX_SPEED = 300;
   private Context _context;
   private Sea _sea;
   private FishSprite _leftSprite;
   private FishSprite _rightSprite;

   private int _direction = -1;
   private int _speedFraction;
   private long _previousTime;


   public SeaAnimal(Context context, Sea sea){
      this._context = context;
      this._sea = sea;
   }

   protected void initialize(Bitmap leftBitmap, Bitmap rightBitmap, int fps, int totalFrames, Point
startPoint, int speed){
      this._leftSprite = new FishSprite(leftBitmap, fps, totalFrames, startPoint);
      this._rightSprite = new FishSprite(rightBitmap, fps, totalFrames, startPoint);
      this._speedFraction = (MAX_SPEED / speed) * 10;
   }

   private FishSprite getSprite(){
      if(this._direction < 0){
         return this._leftSprite;
      }
      return this._rightSprite;
   }

   public int getDirection(){
      FishSprite sprite = this.getSprite();
      int xPos = sprite.getXPos();
      if(this._direction < 0){
         xPos += sprite.getWidth();
      }
      if(xPos < this._sea.getLeft()){
         this._direction = 1;
      }else if(xPos > this._sea.getRight()){
         this._direction = -1;
      }else{
         // Do nothing
      }

      return this._direction;
   }

   public Context getContext(){
      return this._context;
   }

   public Sea getAquarium(){
      return this._sea;
   }

   @Override
   public void render(Canvas canvas){
      long currentTime = System.currentTimeMillis();
      this.getSprite().render(canvas, currentTime);
      this.swim(currentTime);
   }

   public void swim(long currentTime){
      long diff = currentTime - this._previousTime;
      if(diff > this._speedFraction){
         int currentX = this.getSprite().getXPos();
         this.getSprite().setXPos(currentX + this.getDirection());
         this._previousTime = currentTime;
      }
}
```