

自然写互动课堂电视端应用软件 V1.0

软件鉴别材料 — 用户操作手册与设计说明书

软件全称：自然写互动课堂电视端应用软件

软件版本：V1.0

权利人：深圳自然写科技有限公司

文档类型：智能电视应用用户操作手册 + 设计说明书

文档编号：WRITECH-APP-TV-DS-001

编制日期：2026年2月

适用平台：Android TV (Android 9.0+) / 主流智能电视操作系统

目录

- 第一章 软件整体概述
- 第二章 系统架构与设计思路
- 第三章 核心模块功能详细说明
- 第四章 操作流程与使用步骤
- 第五章 与源代码的对应关系
- 附录

第一章 软件整体概述

1.1 软件简介与功能综述

自然写互动课堂电视端应用软件（以下简称"TV APP"）运行于家庭电视或教室电视大屏，是互动课堂系统中面向大屏展示场景的专属客户端。TV APP基于Android TV Leanback框架开发，适配遥控器D-Pad焦点导航交互方式，提供学生书写大屏投射、课堂互动展示、字帖临摹辅助、学情报告浏览等功能。

TV APP的核心使用场景分为两类：1. **教室电视**：配合教师智慧黑板，作为辅助大屏在教室后排展示全班书写状态，让每个角落的学生都能看到作品展示 2. **家庭电视**：家长/学生在家中用电视大屏回放书写过程、查看学情报告、进行字帖临摹练习

主要功能模块：

功能模块	说明
笔迹实时大屏投射	接收网关/算力盒推送的实时笔迹坐标，大屏渲染展示

功能模块	说明
多学生书写同屏对比	最多9宫格展示多名学生的实时书写内容
课堂互动答题展示	大屏展示互动题目，收卷后显示全班答题统计和典型答案
字帖临摹大屏辅助	电视大屏展示放大范字，学生对照练习（课堂或家庭场景）
学情报告大屏浏览	用遥控器浏览孩子学情报告、成绩趋势、薄弱知识点
设备自动发现与连接	通过mDNS自动发现同一局域网内的教室网关，无需手动配置
遥控器/语音操控	适配遥控器D-Pad全程无触屏操作，支持语音搜索

1.2 软件用途与适用场景

场景一：教室辅助大屏（课堂使用）

教室后方电视作为辅助显示屏，实时展示： – 全班书写进度（哪些学生已完成/书写中/未开始） – 教师选取的优秀学生作品放大展示 – 互动答题结果统计饼图和柱状图

场景二：家庭学习辅助（课后使用）

学生用遥控器操作家庭电视： – 打开字帖临摹练习，电视大屏展示标准范字（比手机大10倍以上） – 对照大屏书写，前置摄像头（如有）拍摄书写过程进行实时对比 – 家长陪同查看本周学情报告

场景三：书写成果展示

家庭聚会或亲子活动中，家长投屏孩子的优秀作品，通过回放功能展示孩子的书写成长历程。

1.3 运行环境与系统要求

配置项	最低要求	推荐配置
操作系统	Android TV 9.0（API 28）	Android TV 11.0+
内存	2GB RAM	4GB RAM
存储	500MB可用空间	2GB可用空间
网络	WiFi 802.11n（2.4GHz）	WiFi 6（802.11ax）
分辨率	1920×1080（全高清）	3840×2160（4K）
处理器	ARM Cortex-A53 四核	ARM Cortex-A73 八核
GPU	支持OpenGL ES 3.0	支持Vulkan 1.1

支持的电视品牌/平台： – Android TV（索尼BRAVIA TV、飞利浦Android TV等） – Google TV（Chromecast with Google TV） – 小米/TCL/海信/创维等搭载Android TV的智能电视

1.4 开发语言与技术规范

技术	版本	用途
Kotlin	1.9.0	主要开发语言
Android TV Leanback	1.2.0	TV专用UI框架（焦点导航）
ViewModel + LiveData	Lifecycle 2.7	MVVM架构
OkHttp	4.12.0	HTTP网络请求
WebSocket（OkHttp ws）	4.12.0	实时笔迹数据接收
Room	2.6.1	本地SQLite数据库
Glide	4.16.0	图片加载与缓存
ExoPlayer	2.19.1	视频/动画播放（书写回放）
mDNS（NSD API）	Android NSD	教室网关自动发现
Gson	2.10.1	JSON序列化/反序列化

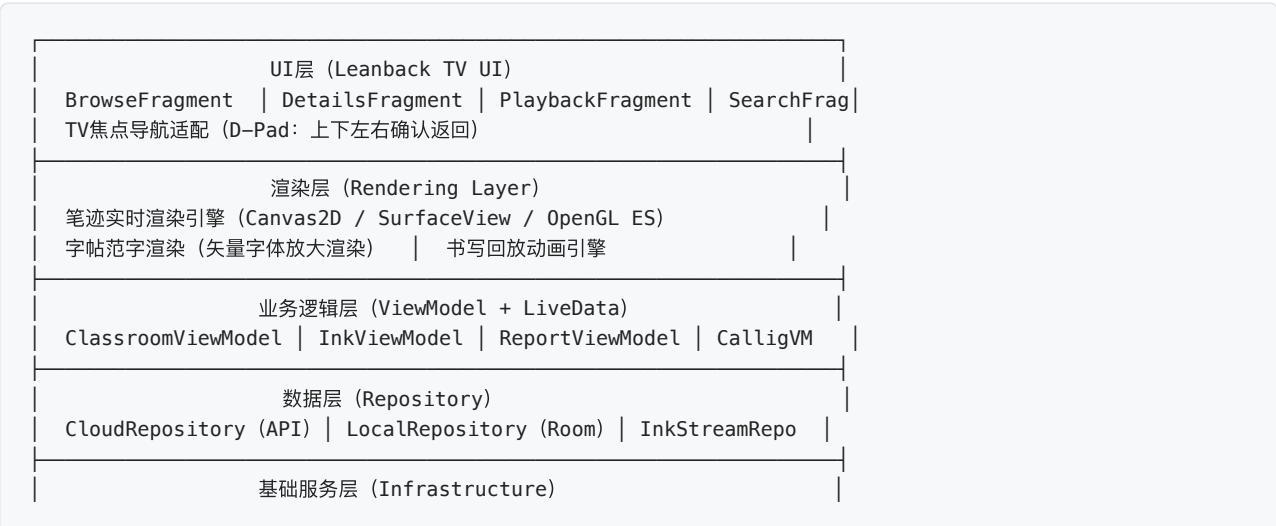
1.5 版本说明

版本	日期	主要变更
V0.7 Beta	2025年9月	基础展示功能，笔迹大屏渲染
V0.9 RC	2025年12月	字帖临摹、学情报告、mDNS设备发现
V1.0	2026年2月	正式版：4K渲染优化、多学生同屏、语音搜索

第二章 系统架构与设计思路

2.1 总体架构设计

TV APP采用MVVM架构，针对大屏电视场景进行了专项优化。应用分为五层：



2.2 Leanback TV UI架构说明

Android TV应用区别于手机APP的核心差异是：**焦点导航（Focus Navigation）**。所有UI交互通过遥控器方向键（D-Pad）控制焦点移动，通过"确认键"触发操作，而非触摸屏点击。

TV APP采用Leanback Library提供的核心Fragment：

Fragment类型	功能	说明
BrowseFragment	浏览主页	左侧导航栏 + 右侧内容区（横向滚动）
DetailsFragment	学情报告详情	顶部主内容 + 下方相关内容
PlaybackFragment	笔迹回放	全屏播放控制（进度条、速度调节）
SearchFragment	语音/文字搜索	字帖搜索、学情查找
ErrorFragment	错误提示	网络断开、设备未找到等错误

焦点导航逻辑（TV D-Pad适配）：

```
// tv/ui/classroom/InkDisplayFragment.kt
class InkDisplayFragment : Fragment() {

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        // 配置D-Pad焦点导航
        // 主要功能区域（笔迹展示）作为默认焦点
        binding.inkDisplayArea.requestFocus()

        // 底部控制栏：方向键上进入笔迹展示，方向键下进入控制栏
        binding.controlPanel.setOnFocusChangeListener { _, hasFocus ->
            binding.controlBar.visibility =
                if (hasFocus) View.VISIBLE else View.GONE
        }

        // 遥控器按键处理
        view.setOnKeyListener { _, keyCode, event ->
            if (event.action == KeyEvent.ACTION_DOWN) {
                when (keyCode) {
                    KeyEvent.KEYCODE_DPAD_CENTER,
                    KeyEvent.KEYCODE_ENTER -> {
                        // 确认键：选中当前学生作品进行放大展示
                        viewModel.selectCurrentFocusedStudent()
                        true
                    }
                    KeyEvent.KEYCODE_BACK -> {
                        // 返回键：退出全屏，返回多人视图
                        if (viewModel.isFullscreen.value == true) {
                            viewModel.exitFullscreen()
                            true
                        } else false
                    }
                }
            } else -> false
        }
    }
}
```

```
        } else false
    }
}
}
```

2.3 笔迹渲染引擎设计

大屏笔迹渲染是TV APP的核心技术挑战：需要在4K电视屏幕上以60fps流畅渲染多个学生的实时笔迹，同时保持渲染质量（线条平滑、无锯齿）。

渲染方案对比与选择：

渲染方案	优点	缺点	使用场景
Canvas 2D	简单，适合少量线段	大量线段时性能差	单学生笔迹回放
SurfaceView	独立渲染线程，不卡UI	与UI层混合复杂	实时笔迹渲染（多学生）
OpenGL ES	最高性能，支持GPU加速	开发复杂	高密度多学生4K渲染

TV APP为多学生同屏场景采用SurfaceView + 独立渲染线程方案：

```
// tv/ui/rendering/InkSurfaceView.kt
class InkSurfaceView(context: Context) : SurfaceView(context),
    SurfaceHolder.Callback {

    private val renderThread = HandlerThread("InkRenderThread")
    private lateinit var renderHandler: Handler

    // 每个学生的笔迹缓存（学生ID -> 笔画列表）
    private val studentInkMap = ConcurrentHashMap<String, MutableList<StrokePath>>()

    // 渲染帧率控制（目标60fps）
    private var lastRenderTime = 0L
    private val targetFrameInterval = 1000L / 60L // 约16.7ms

    override fun surfaceCreated(holder: SurfaceHolder) {
        renderThread.start()
        renderHandler = Handler(renderThread.looper)
        scheduleNextFrame()
    }

    private fun scheduleNextFrame() {
        val now = SystemClock.uptimeMillis()
        val delay = maxOf(0, targetFrameInterval - (now - lastRenderTime))
        renderHandler.postDelayed({ renderFrame() }, delay)
    }

    private fun renderFrame() {
        val canvas = holder.lockCanvas() ?: return
        try {
            // 清除背景（白色）
            canvas.drawColor(Color.WHITE)

            // 渲染所有学生的笔迹
            val paint = Paint().apply {
                style = Paint.Style.STROKE
                strokeCap = Paint.Cap.ROUND
                strokeJoin = Paint.Join.ROUND
            }
            studentInkMap.forEach { (studentId, strokes) -> {
                strokes.forEach { stroke -> {
                    stroke.draw(canvas, paint)
                }
            }
        }
        }
        holder.unlockCanvasAndPost(canvas)
    }
}
```

```

        isAntiAlias = true
    }

    studentInkMap.forEach { (studentId, strokes) ->
        val color = getStudentColor(studentId)
        paint.color = color
        paint.strokeWidth = 6f // TV大屏适配: 线宽增大

        strokes.forEach { stroke ->
            _drawStrokeWithBezier(canvas, stroke.points, paint,
                                   canvas.width, canvas.height)
        }
    }

    } finally {
        holder.unlockCanvasAndPost(canvas)
        lastRenderTime = SystemClock.uptimeMillis()
        scheduleNextFrame()
    }
}

fun addInkPoint(studentId: String, point: StrokePoint) {
    renderHandler.post {
        val strokes = studentInkMap.getOrPut(studentId) { mutableListOf() }
        if (point.penUp && strokes.isNotEmpty()) {
            strokes.last().isComplete = true
        } else if (!point.penUp) {
            if (strokes.isEmpty() || strokes.last().isComplete) {
                strokes.add(StrokePath(mutableListOf(point)))
            } else {
                strokes.last().points.add(point)
            }
        }
    }
}
}

```

2.4 数据设计

Room数据库表（本地缓存）：

```

// tv/data/database/entities.kt

@Entity(tableName = "classroom_ink_cache")
data class InkCacheEntity(
    @PrimaryKey val id: String,
    val classroom_id: String,
    val student_id: String,
    val student_name: String,
    val ink_json: String,           // 笔迹坐标JSON序列化
    val created_at: Long,
    val is_realtime: Int           // 1=实时缓存, 0=历史数据
)

@Entity(tableName = "calligraphy_templates")
data class CalligraphyTemplate(
    @PrimaryKey val id: String,
    val title: String,
    val subject: String,
    val grade: String,
    val characters: String,        // 字帖中的汉字列表（逗号分隔）
)

```

```

        val resource_url: String,        // CDN资源URL
        val local_path: String?,         // 本地缓存路径（下载后）
        val cached_at: Long?
    )

@Entity(tableName = "bound_devices")
data class BoundDevice(
    @PrimaryKey val device_id: String,
    val device_type: String,             // "gateway" / "edge_box"
    val device_name: String,
    val ip_address: String,
    val port: Int,
    val school_id: String,
    val last_seen: Long
)

```

2.5 接口设计

WebSocket实时数据接收:

```

// tv/data/network/InkWebSocketClient.kt
class InkWebSocketClient(private val serverUrl: String) {

    private val client = OkHttpClient.Builder()
        .readTimeout(0, TimeUnit.MILLISECONDS) // 长连接无超时
        .build()

    private var websocket: WebSocket? = null
    private val _inkFlow = MutableSharedFlow<StudentInkData>()
    val inkFlow: SharedFlow<StudentInkData> = _inkFlow

    fun connect(classroomId: String, token: String) {
        val request = Request.Builder()
            .url("$serverUrl/ws/v1/ink?classroom_id=$classroomId")
            .header("Authorization", "Bearer $token")
            .build()

        websocket = client.newWebSocket(request, object : WebSocketListener() {
            override fun onMessage(webSocket: WebSocket, text: String) {
                val data = parseInkMessage(text)
                if (data != null) {
                    // 在协程作用域发射到Flow
                    scope.launch { _inkFlow.emit(data) }
                }
            }

            override fun onFailure(webSocket: WebSocket,
                                    t: Throwable, response: Response?) {
                // 5秒后自动重连
                scope.launch {
                    delay(5000)
                    connect(classroomId, token)
                }
            }
        })
    }

    private fun parseInkMessage(text: String): StudentInkData? {
        return try {
            val json = JSONObject(text)
            if (json.getString("type") != "stroke.realtime") return null
        } catch (e: Exception) {
            null
        }
    }
}

```

```

        StudentInkData(
            studentId = json.getString("student_id"),
            studentName = json.getString("student_name"),
            points = parsePoints(json.getJSONArray("points")),
            timestamp = json.getLong("timestamp")
        )
    } catch (e: Exception) { null }
}
}

```

mDNS设备自动发现:

```

// tv/data/network/GatewayDiscovery.kt
class GatewayDiscovery(private val context: Context) {

    private val nsdManager = context.getSystemService(Context.NSD_SERVICE)
        as NsdManager
    private val discoveredGateways = mutableListOf<GatewayInfo>()

    fun startDiscovery(onFound: (GatewayInfo) -> Unit) {
        val listener = object : NsdManager.DiscoveryListener {
            override fun onServiceFound(serviceInfo: NsdServiceInfo) {
                // 发现 _writech-gateway._tcp 类型的服务
                if (serviceInfo.serviceType == "_writech-gateway._tcp.") {
                    // 解析服务详细信息
                    nsdManager.resolveService(serviceInfo,
                        object : NsdManager.ResolveListener {
                            override fun onServiceResolved(service: NsdServiceInfo) {
                                val gateway = GatewayInfo(
                                    name = service.serviceName,
                                    host = service.host.hostAddress,
                                    port = service.port,
                                    schoolId = service.attributes["school_id"]
                                        ?.let { String(it) }
                                )
                                discoveredGateways.add(gateway)
                                onFound(gateway)
                            }
                        })
                    override fun onResolveFailed(si: NsdServiceInfo, ec: Int) {}
                })
            }
        }
        override fun onDiscoveryStarted(serviceType: String) {}
        override fun onDiscoveryStopped(serviceType: String) {}
        override fun onServiceLost(serviceInfo: NsdServiceInfo) {
            discoveredGateways.removeAll {
                it.name == serviceInfo.serviceName
            }
        }
        override fun onStartDiscoveryFailed(st: String, ec: Int) {}
        override fun onStopDiscoveryFailed(st: String, ec: Int) {}
    }

    nsdManager.discoverServices("_writech-gateway._tcp.",
        NsdManager.PROTOCOL_DNS_SD,
        listener)
}
}

```

2.6 安全设计

- **设备认证**：TV APP通过API Token认证（设备首次绑定时在手机APP上扫码授权）
- **内容保护**：课堂展示内容显示时启用 `FLAG_SECURE`（禁止系统截屏录屏）
- **Token存储**：存储于Android EncryptedSharedPreferences（KeyStore加密）
- **局域网隔离**：仅接受来自同一局域网段（同一WiFi）内的网关WebSocket连接
- **防截屏**：课堂内容展示期间设置 `window.addFlags(FLAG_SECURE)` 防止录屏

2.7 字帖渲染设计

字帖大屏临摹功能需要将汉字字形以高质量矢量方式放大展示（最大放满4K屏幕的1/4区域，约960×960像素）：

```
// tv/ui/calligraphy/CalligraphyDisplayView.kt
class CalligraphyDisplayView(context: Context) : View(context) {

    // 使用矢量字体 (TrueType)，避免位图放大失真
    private var characterPath: Path? = null
    private val strokePaint = Paint().apply {
        style = Paint.Style.STROKE
        strokeWidth = 8f
        color = Color.RED           // 描红帖：红色范字
        isAntiAlias = true
    }

    // 加载字符的矢量轮廓路径（从字体文件解析）
    fun setCharacter(char: Char) {
        // 使用Android字体API获取字形路径
        val paint = Paint().apply {
            textSize = 200f
            typeface = ResourcesCompat.getFont(context, R.font.kaishu)
        }
        characterPath = Path().also { paint.getTextPath(char.toString(),
            0, 1, 0f, 200f, it) }

        invalidate()
    }

    override fun onDraw(canvas: Canvas) {
        super.onDraw(canvas)

        // 绘制田字格背景辅助线
        drawGridHelper(canvas)

        // 绘制字形（缩放到适合大屏展示的大小）
        characterPath?.let { path ->
            val scaleMatrix = Matrix()
            val scale = (width * 0.8f) / 200f // 缩放到屏幕宽度80%
            scaleMatrix.setScale(scale, scale, width / 2f, height / 2f)
            val scaledPath = Path(path).apply { transform(scaleMatrix) }
            canvas.drawPath(scaledPath, strokePaint)
        }
    }

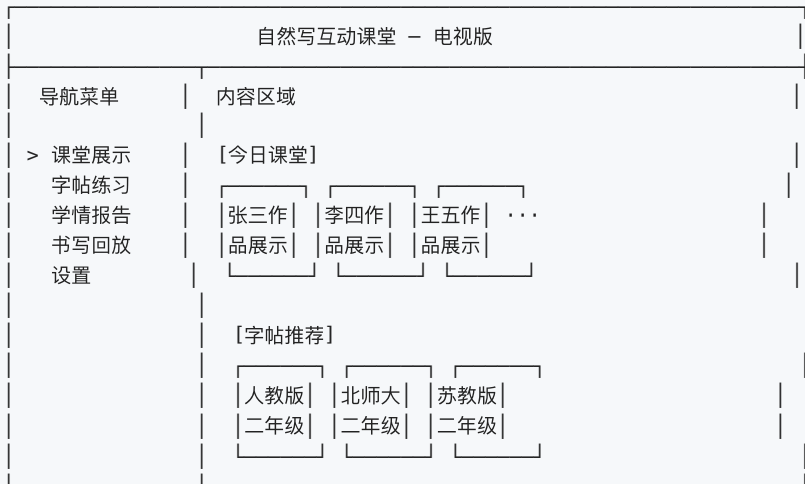
    private fun drawGridHelper(canvas: Canvas) {
        val gridPaint = Paint().apply {
            color = Color.parseColor("#CCCCCC")
            strokeWidth = 2f
        }
        // 绘制田字格
        canvas.drawLine(width / 2f, 0f, width / 2f, height.toFloat(), gridPaint)
        canvas.drawLine(0f, height / 2f, width.toFloat(), height / 2f, gridPaint)
    }
}
```

```
// 绘制外框
canvas.drawRect(40f, 40f, width - 40f, height - 40f, gridPaint)
}
}
```

第三章 核心模块功能详细说明

3.1 主页浏览模块（BrowseFragment）

TV APP主页采用Leanback BrowseFragment布局，左侧为功能导航菜单，右侧为内容卡片列表：



BrowseFragment配置：

```
// tv/ui/main/MainBrowseFragment.kt
class MainBrowseFragment : BrowseFragment() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // TV主题配置
        headersState = HEADERS_ENABLED
        isHeadersTransitionOnBackEnabled = true

        // 品牌颜色
        brandColor = ContextCompat.getColor(requireContext(), R.color.writech_blue)
        title = "自然写互动课堂"
    }

    private fun setupRows() {
        val rowsAdapter = ArrayObjectAdapter(ListRowPresenter())

        // 课堂展示行
        val classroomRow = buildClassroomRow()
        rowsAdapter.add(classroomRow)

        // 字帖推荐行
        val calligraphyRow = buildCalligraphyRow()
        rowsAdapter.add(calligraphyRow)
    }
}
```

```
// 孩子学情行
val reportRow = buildReportRow()
rowsAdapter.add(reportRow)

adapter = rowsAdapter
}
```

3.2 实时笔迹大屏展示模块

多学生同屏展示（九宫格布局）：

课堂实时书写 - 二年级一班 已连接：38支笔		
张三 [书写中] [笔迹图]	李四 [已完成] [笔迹图]	王五 [书写中] [笔迹图]
赵六 [书写中] [笔迹图]	陈七 [未开始] (空白)	周八 [书写中] [笔迹图]
吴九 [已完成] [笔迹图]	... (+29人)	[查看全部]

单学生作品全屏展示（按确认键放大）：

张三的作答 - 第5课生字练习	[×] 关闭
[学生书写笔迹全屏大图显示]	
一 大 天 地 水 火 山 石	
AI评分：92分 笔顺正确率：96% 书写规范度：88%	
[上一个学生 ◀]	[书写回放 ▶] [▶ 下一个学生]

3.3 字帖临摹大屏辅助模块

字帖展示界面（教学场景）：

人教版二年级上册 - 第5课生字 [×] [◀上一字] [下一字▶]	
范字（电视左半屏展示）	学生实时书写（右半屏）
美 (红色描红楷书体)	[学生书写笔迹实时显示]

笔顺演示：▶ 播放

笔顺：9笔 ✓ (正确书写中)
规范度：87%

3.4 互动答题结果展示模块

教师在PC或黑板端发题、收卷后，TV APP接收到结果推送，以大屏动画形式展示统计数据：

答题结果 - 题目：2+3=?

全班作答情况 (38人)

正确：30人(79%)
错误：8人(21%)

典型正确答案：

[张三：5] [李四：5] [王五：5]

典型错误答案：

[陈七：4] [周八：6] - 教师：重点讲解加法概念

3.5 学情报告大屏浏览模块

家长用遥控器浏览孩子学情报告 (Leanback DetailsFragment)：

张小明的学情报告 - 本周 (2/10-2/14)

本周总结

- 完成作业：5/5 ✓
- 平均得分：88.5分
- 书写规范：↑提升3%
- 笔顺正确：92%

薄弱知识点：

△ 数学应用题 △ 多音字辨析

成长轨迹图

分数

100 | *
90 | * *
80 | * * *
70 |
第1 2 3 4 5周

本周优秀作品 [浏览 ▶]

第一 | 第二 | 第三 |
次作 | 次作 | 次作 |

3.6 设备绑定与设置模块

设备绑定流程 (首次使用)：

TV APP首次打开：

显示"扫码绑定"界面

请用手机APP扫描此二维码
完成设备绑定
[二维码图案 (128×128)]
二维码有效期：5分钟

家长/教师在手机APP中扫描
→ 手机APP发送授权请求到云端
→ 云端验证并绑定TV设备ID

TV APP收到绑定成功通知
→ 自动登录，跳转主页

第四章 操作流程与使用步骤

4.1 安装与首次启动

通过应用商店安装： 1. 在电视遥控器按主页键，进入电视应用商店 2. 搜索"自然写互动课堂"或"Writech" 3. 选择下载安装（约120MB） 4. 安装完成后从应用列表启动

通过U盘旁加载（支持离线安装）： 1. 将APK文件复制到U盘 2. 在电视文件管理器中找到APK文件，点击安装 3. 允许"安装未知来源应用"（首次安装需开启此选项）

4.2 基本遥控器操作说明

遥控器按键	功能
方向键（上/下/左/右）	移动焦点
确认键 / OK	选中当前元素（进入/播放/展开）
返回键	返回上一页
主页键	返回电视主页（后台运行APP）
菜单键	打开当前页面的更多选项
语音键（如有）	唤起语音搜索
数字键（0-9）	快速输入数字（设置页面使用）

4.3 课堂展示使用流程（教室TV场景）

操作步骤：

1. 开启课堂前，教室TV开机，APP自动发现教室网关（mDNS）

2. 教师在黑板或PC端开始课堂，TV APP收到WebSocket通知自动进入课堂模式

3. 大屏自动切换到"实时书写"界面，展示学生书写状态

4. 教师按遥控器方向键选择学生，按确认键放大查看

5. 教师在黑板端收卷后，TV大屏自动展示答题统计结果（动画呈现）

6. 课堂结束，TV APP退出课堂模式，显示"课堂已结束"提示

4.4 字帖练习使用流程（家庭TV场景）

- 操作步骤：
- 1. 打开APP，在主页选择"字帖练习"
 - 2. 按方向键选择年级（一年级/二年级/...）
 - 3. 选择课本版本（人教版/北师大版/苏教版）
 - 4. 选择本周字帖内容（APP自动推荐与作业关联的字帖）
 - 5. 字帖内容加载，大屏左侧显示范字，右侧为学生书写区
 - 6. 学生手持点阵笔在字帖纸上书写，实时书写内容显示在电视右半屏
 - 7. 每个字书写完成后，AI给出笔顺和规范度评分
 - 8. 家长用遥控器翻页到下一个字

4.5 学情报告浏览流程

- 操作步骤：
- 1. 在主页选择"学情报告"
 - 2. 选择孩子（多孩子家庭可切换）
 - 3. 按方向键选择报告类型（周报/月报/学期报）
 - 4. 用方向键上下滚动浏览报告内容
 - 5. 方向键右进入"成长轨迹"图表（可交互查看每周数据）
 - 6. 按菜单键可选择"分享"（截图发给亲戚）或"收藏"

4.6 连接问题排查

问题	排查步骤
找不到教室网关	① 确认TV与网关在同一WiFi ② 关闭再开启WiFi ③ 手动输入网关IP
实时笔迹卡顿	① 检查WiFi信号强度 ② 将TV通过网线连接路由器 ③ 降低并发展示学生数量
二维码扫描失败	① 确保手机APP版本≥V1.0 ② 二维码5分钟有效，刷新后重试

第五章 与源代码的对应关系

5.1 模块与源代码文件对应表

功能模块	源代码路径	说明
主页浏览	tv/ui/main/MainBrowseFragment.kt	Leanback BrowseFragment主界面
焦点导航配置	tv/ui/main/FocusNavigationManager.kt	D-Pad焦点路由规则
笔迹实时展示	tv/ui/classroom/InkDisplayFragment.kt	实时笔迹大屏展示Fragment
笔迹渲染引擎	tv/ui/rendering/InkSurfaceView.kt	SurfaceView多学生并发渲染
多学生同屏	tv/ui/classroom/MultiStudentGridView.kt	九宫格学生作品展示
字帖临摹模块	tv/ui/calligraphy/CalligraphyFragment.kt	字帖大屏展示与临摹辅助
字形渲染	tv/ui/calligraphy/CalligraphyDisplayView.kt	矢量字形放大渲染

功能模块	源代码路径	说明
答题结果展示	tv/ui/classroom/QuizResultFragment.kt	互动答题统计大屏展示
学情报告	tv/ui/report/ReportDetailsFragment.kt	Leanback DetailsFragment学情报告
书写回放	tv/ui/replay/StrokePlaybackFragment.kt	Leanback PlaybackFragment书写回放
设备发现	tv/data/network/GatewayDiscovery.kt	mDNS教室网关自动发现
WebSocket客户端	tv/data/network/InkWebSocketClient.kt	实时笔迹数据流接收
设备绑定	tv/ui/binding/DeviceBindingFragment.kt	二维码扫码绑定流程
本地数据库	tv/data/database/	Room数据库实体与DAO
主题配置	tv/ui/theme/TVTheme.kt	TV大屏专用视觉主题

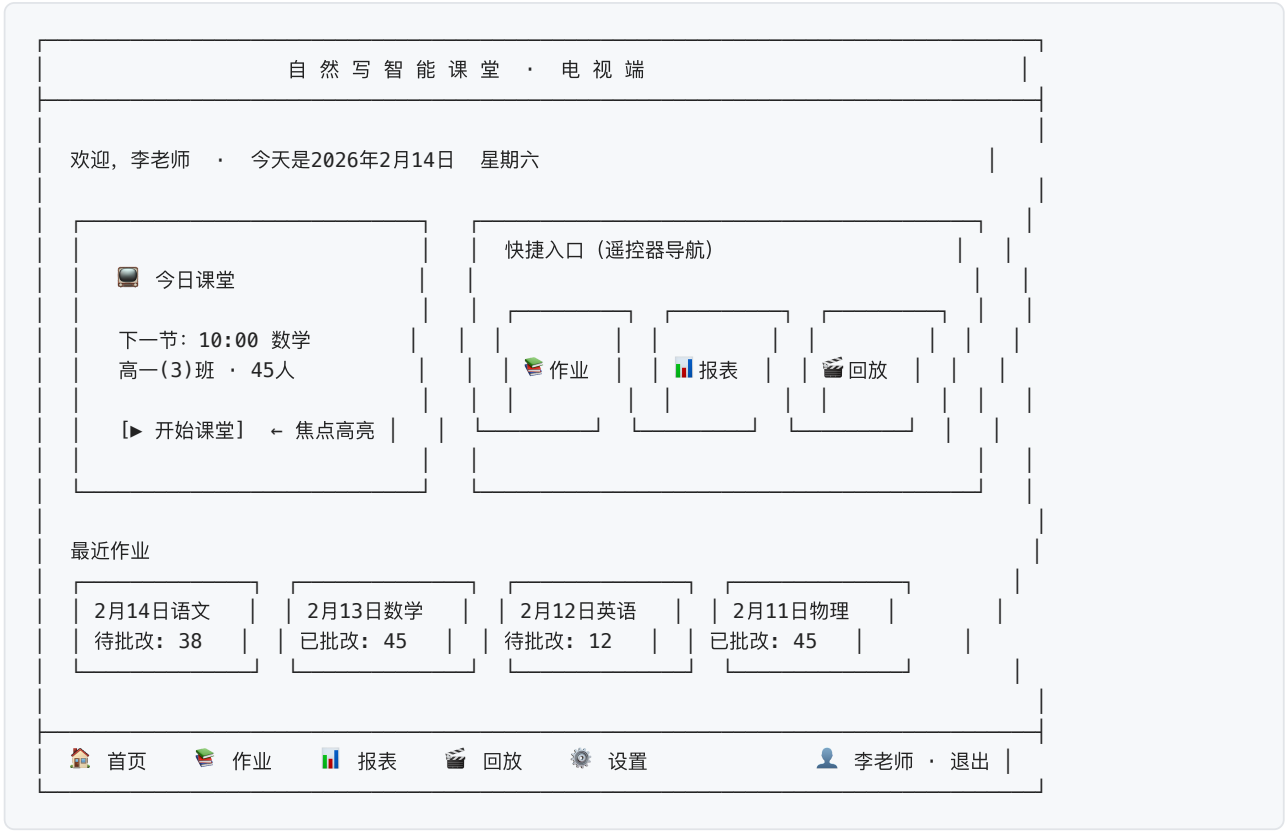
5.2 核心类与方法说明

类名	所在文件	功能说明
MainBrowseFragment	main/MainBrowseFragment.kt	TV应用主界面，Leanback浏览体验
InkSurfaceView	rendering/InkSurfaceView.kt	多学生笔迹并发渲染 (SurfaceView)
CalligraphyDisplayView	calligraphy/CalligraphyDisplayView.kt	汉字矢量放大渲染，田字格辅助
InkWebSocketClient	network/InkWebSocketClient.kt	WebSocket笔迹流接收与解析
GatewayDiscovery	network/GatewayDiscovery.kt	Android NSD mDNS网关发现
ClassroomViewModel	viewmodel/ClassroomViewModel.kt	课堂展示状态管理
InkViewModel	viewmodel/InkViewModel.kt	实时笔迹数据处理
ReportViewModel	viewmodel/ReportViewModel.kt	学情报告数据管理

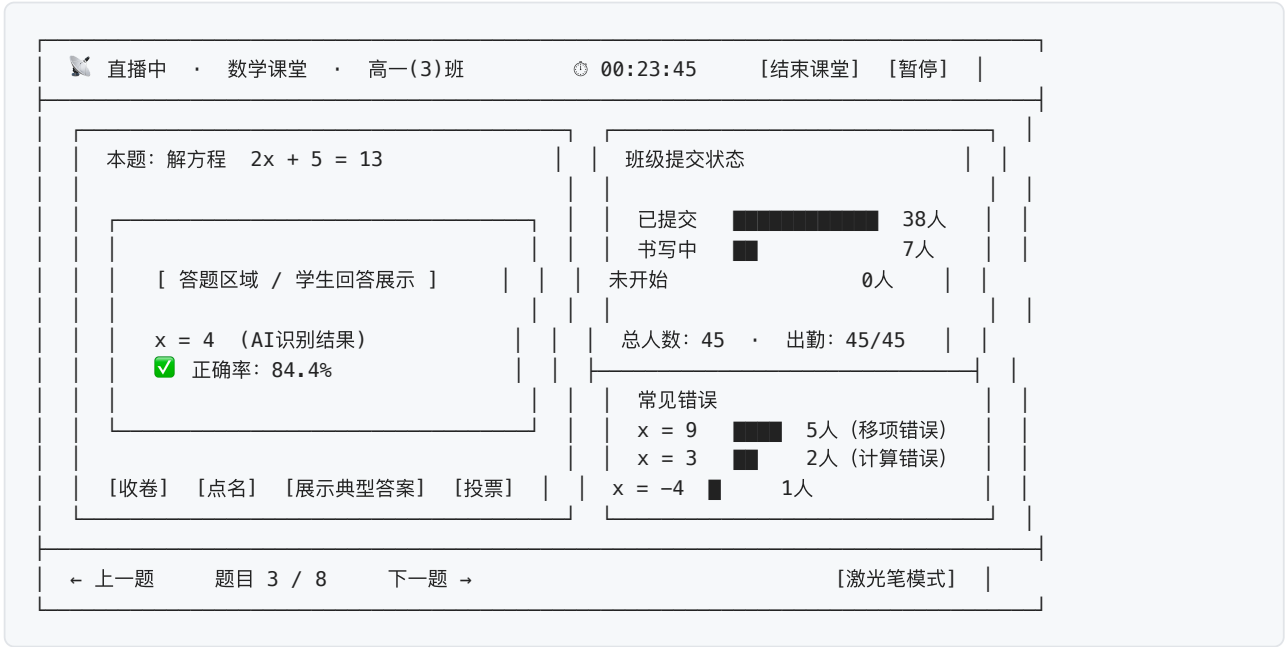
附录A 界面设计稿 (GUI Mockup)

本附录以TV横屏线框图形式呈现电视APP各核心界面的设计稿，遵循10英尺UI设计规范（适配2米以上观看距离）。

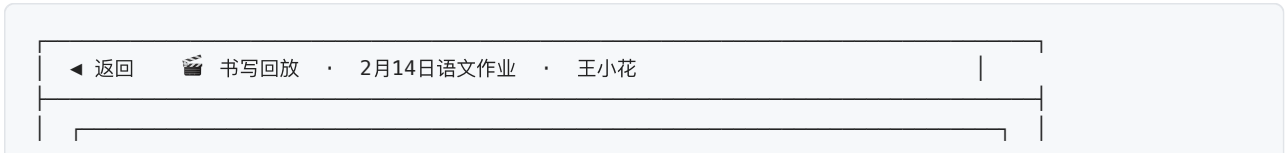
A.1 应用首页 (Home)

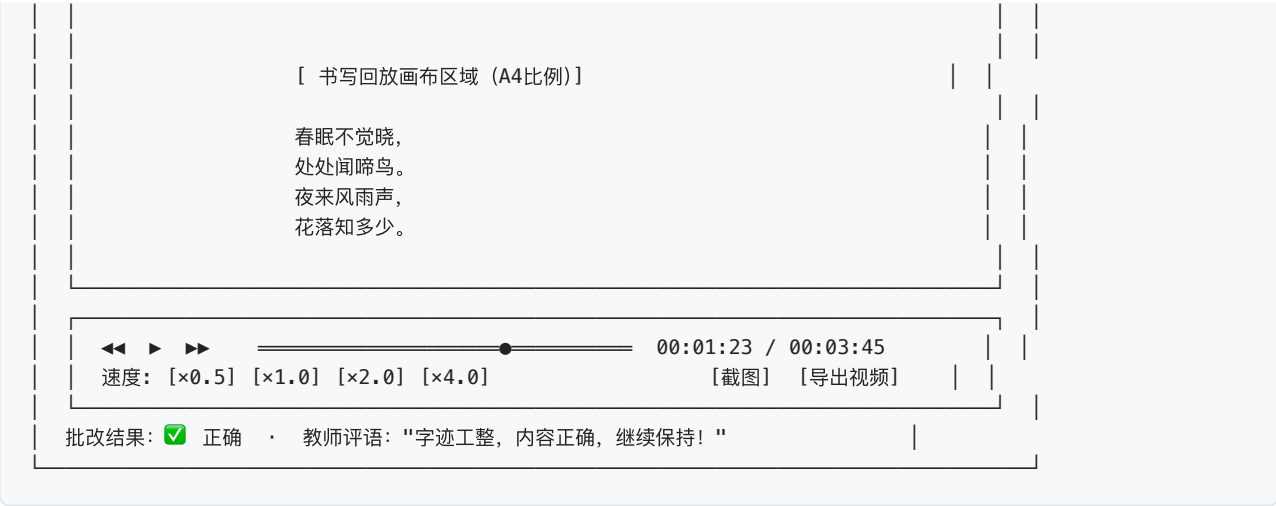


A.2 课堂互动界面 (教师投屏大屏)

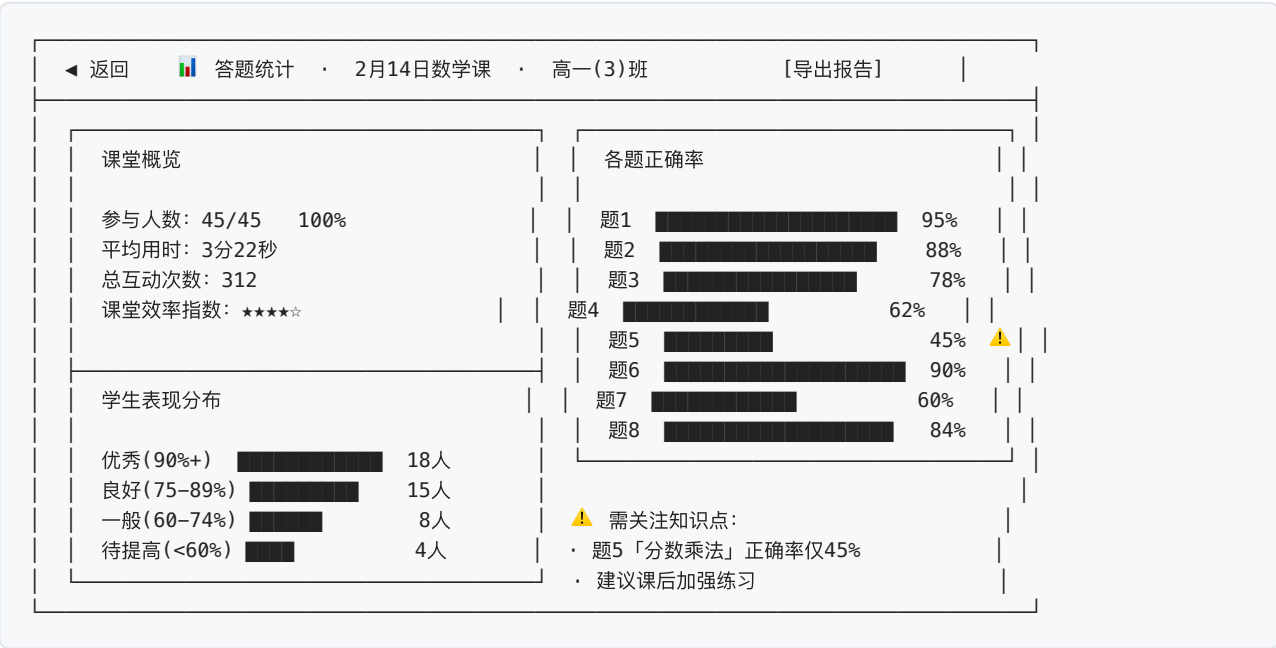


A.3 书写回放界面





A.4 答题统计报告界面



附录B 遥控器按键映射表

Android KeyCode	遥控器按键	TV APP行为
KEYCODE_DPAD_UP	方向↑	焦点上移
KEYCODE_DPAD_DOWN	方向↓	焦点下移
KEYCODE_DPAD_LEFT	方向←	焦点左移 / 上一项
KEYCODE_DPAD_RIGHT	方向→	焦点右移 / 下一项
KEYCODE_DPAD_CENTER	确认	选中/播放/展开
KEYCODE_BACK	返回	退出当前页面

Android KeyCode	遥控器按键	TV APP行为
KEYCODE_HOME	主页	挂起APP，返回TV主页
KEYCODE_MENU	菜单	展开更多选项
KEYCODE_MEDIA_PLAY_PAUSE	播放/暂停	书写回放播放控制
KEYCODE_MEDIA_FAST_FORWARD	快进	回放速度加快
KEYCODE_MEDIA_REWIND	快退	回放进度后退

附录B 术语表

术语	说明
Android TV	Android系统的TV版本，适配大屏遥控器操作
Leanback Library	Android TV官方UI框架，提供BrowseFragment等TV专用组件
D-Pad	Directional Pad，遥控器方向键（上下左右确认）
焦点导航	TV应用中通过方向键移动"焦点"来操作UI的交互方式
NSD	Network Service Discovery，Android mDNS实现
mDNS	Multicast DNS，局域网内零配置服务发现协议
SurfaceView	Android高性能绘图组件，拥有独立渲染线程（适合游戏/视频/实时笔迹）
FLAG_SECURE	Android窗口标志，设置后禁止系统截图和录屏

文档编制：深圳自然写科技有限公司 Android TV研发团队

文档版本：V1.0

最后更新：2026年2月14日

版权所有 © 2026 深圳自然写科技有限公司

附录C 核心技术实现详述

C.1 SurfaceView双缓冲渲染架构

Android TV应用中，笔迹实时渲染采用SurfaceView双缓冲机制。主线程负责业务逻辑，独立渲染线程（RenderThread）负责Canvas绘图，避免UI阻塞。

C.1.1 渲染线程设计

```
// TvInkSurfaceView.java - 双缓冲渲染核心实现
public class TvInkSurfaceView extends SurfaceView implements SurfaceHolder.Callback {

    private static final String TAG = "TvInkSurfaceView";
    private static final int TARGET_FPS = 60;
    private static final long FRAME_INTERVAL_NS = 1_000_000_000L / TARGET_FPS;

    private RenderThread mRenderThread;
    private final Object mLock = new Object();
    private volatile boolean mRunning = false;

    // 双缓冲: 前台缓冲 (显示) 和后台缓冲 (绘制)
    private Bitmap mFrontBuffer;
    private Bitmap mBackBuffer;
    private Canvas mBackCanvas;

    // 笔迹数据队列 (生产者-消费者模型)
    private final ConcurrentLinkedQueue<InkPacket> mInkQueue = new ConcurrentLinkedQueue<>();

    // 所有学生笔迹路径缓存 key=studentId
    private final ConcurrentHashMap<String, StudentInkState> mStudentInks = new ConcurrentHashMap<>();

    public TvInkSurfaceView(Context context) {
        super(context);
        getHolder().addCallback(this);
        setZOrderMediaOverlay(true);
    }

    @Override
    public void surfaceCreated(@NonNull SurfaceHolder holder) {
        int width = getWidth();
        int height = getHeight();
        mFrontBuffer = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
        mBackBuffer = Bitmap.createBitmap(width, height, Bitmap.Config.ARGB_8888);
        mBackCanvas = new Canvas(mBackBuffer);
        mRunning = true;
        mRenderThread = new RenderThread();
        mRenderThread.start();
    }

    @Override
    public void surfaceDestroyed(@NonNull SurfaceHolder holder) {
        mRunning = false;
        try {
            mRenderThread.join(2000);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }

    /**
     * 渲染线程主循环
     * 以60fps稳定帧率绘制所有学生笔迹
     */
    private class RenderThread extends Thread {
        @Override
        public void run() {
            long lastFrameTime = System.nanoTime();
            while (mRunning) {
                long now = System.nanoTime();
                long elapsed = now - lastFrameTime;
                if (elapsed < FRAME_INTERVAL_NS) {

```

```

        try {
            Thread.sleep((FRAME_INTERVAL_NS - elapsed) / 1_000_000);
        } catch (InterruptedException ignored) {}
        continue;
    }
    lastFrameTime = now;
    processInkQueue();
    drawFrame();
    swapBuffers();
}

/** 消费队列中的笔迹数据包, 更新各学生的Path */
private void processInkQueue() {
    InkPacket packet;
    while ((packet = mInkQueue.poll()) != null) {
        StudentInkState state = mStudentInks.computeIfAbsent(
            packet.studentId, k -> new StudentInkState(k));
        state.addPoint(packet.x, packet.y, packet.pressure, packet.isPenUp);
    }
}

/** 将所有学生笔迹绘制到后台缓冲区 */
private void drawFrame() {
    mBackCanvas.drawColor(Color.WHITE);
    for (StudentInkState state : mStudentInks.values()) {
        mBackCanvas.drawPath(state.getCurrentPath(), state.getPaint());
        for (Path historicalPath : state.getHistoricalPaths()) {
            mBackCanvas.drawPath(historicalPath, state.getPaint());
        }
    }
}

/** 交换缓冲区并提交到SurfaceHolder */
private void swapBuffers() {
    synchronized (mLock) {
        Bitmap temp = mFrontBuffer;
        mFrontBuffer = mBackBuffer;
        mBackBuffer = temp;
        mBackCanvas = new Canvas(mBackBuffer);
    }
    Canvas canvas = getHolder().lockCanvas();
    if (canvas != null) {
        synchronized (mLock) {
            canvas.drawBitmap(mFrontBuffer, 0, 0, null);
        }
        getHolder().unlockCanvasAndPost(canvas);
    }
}

/** 外部调用: 向渲染队列投递笔迹数据包 */
public void pushInkPacket(InkPacket packet) {
    mInkQueue.offer(packet);
}

/** 清除指定学生笔迹 */
public void clearStudentInk(String studentId) {
    mStudentInks.remove(studentId);
}

/** 清除全部笔迹 */
public void clearAllInk() {
    mStudentInks.clear();
}

```

```
}  
}
```

C.1.2 学生笔迹状态管理

```
// StudentInkState.java - 单学生笔迹状态  
public class StudentInkState {  
  
    private static final float BASE_STROKE_WIDTH = 4.0f;  
    private static final float PRESSURE_SCALE = 3.0f;  
  
    private final String studentId;  
    private final Paint paint;  
    private Path currentPath;  
    private final List<Path> historicalPaths = new ArrayList<>();  
  
    // 最近两个点, 用于贝塞尔平滑  
    private float prevX = -1, prevY = -1;  
    private float prevMidX, prevMidY;  
  
    public StudentInkState(String studentId) {  
        this.studentId = studentId;  
        this.paint = new Paint(Paint.ANTI_ALIAS_FLAG);  
        this.paint.setStyle(Paint.Style.STROKE);  
        this.paint.setStrokeCap(Paint.Cap.ROUND);  
        this.paint.setStrokeJoin(Paint.Join.ROUND);  
        // 根据studentId哈希分配不同颜色 (最多30种颜色)  
        this.paint.setColor(StudentColorPalette.getColor(studentId));  
        this.currentPath = new Path();  
    }  
  
    /**  
     * 添加笔迹点, 使用二次贝塞尔曲线平滑  
     * @param x      归一化x坐标 [0,1]  
     * @param y      归一化y坐标 [0,1]  
     * @param pressure 压力值 [0,1]  
     * @param isPenUp 抬笔标志  
     */  
    public void addPoint(float x, float y, float pressure, boolean isPenUp) {  
        // 将归一化坐标转换为屏幕像素坐标 (此处假设渲染尺寸已注入)  
        float screenX = x * RenderConfig.CANVAS_WIDTH;  
        float screenY = y * RenderConfig.CANVAS_HEIGHT;  
  
        if (isPenUp) {  
            // 抬笔: 保存当前笔画, 开始新路径  
            if (currentPath != null && !currentPath.isEmpty()) {  
                historicalPaths.add(currentPath);  
                if (historicalPaths.size() > 500) {  
                    historicalPaths.remove(0); // 防止内存溢出  
                }  
            }  
            currentPath = new Path();  
            prevX = -1;  
            prevY = -1;  
            return;  
        }  
  
        float strokeWidth = BASE_STROKE_WIDTH + pressure * PRESSURE_SCALE;  
        paint.setStrokeWidth(strokeWidth);  
  
        if (prevX < 0) {  
            // 第一个点: 直接moveTo
```

```

        currentPath.moveTo(screenX, screenY);
    } else {
        // 后续点：通过中点进行贝塞尔平滑
        float midX = (prevX + screenX) * 0.5f;
        float midY = (prevY + screenY) * 0.5f;
        currentPath.quadTo(prevX, prevY, midX, midY);
    }
    prevX = screenX;
    prevY = screenY;
}

public Path getCurrentPath() { return currentPath; }
public List<Path> getHistoricalPaths() { return historicalPaths; }
public Paint getPaint() { return paint; }
}

```

C.2 mDNS网关自动发现实现

Android TV通过NSD（Network Service Discovery）实现局域网内网关设备的自动发现，无需手动配置IP地址。

C.2.1 NSD服务发现核心代码

```

// TvGatewayDiscoveryManager.java
public class TvGatewayDiscoveryManager {

    private static final String SERVICE_TYPE = "_writech._tcp.";
    private static final String SERVICE_NAME_PREFIX = "WritechGateway-";
    private static final int DISCOVERY_TIMEOUT_MS = 30_000;

    private final NsdManager mNsdManager;
    private NsdManager.DiscoveryListener mDiscoveryListener;
    private NsdManager.ResolveListener mResolveListener;

    // 发现到的网关列表 (ip -> GatewayInfo)
    private final ConcurrentHashMap<String, GatewayInfo> mGateways = new ConcurrentHashMap<>();
    private final List<OnGatewayDiscoveredListener> mListeners = new CopyOnWriteArrayList<>();

    private volatile boolean mDiscovering = false;
    private final Handler mTimeoutHandler = new Handler(Looper.getMainLooper());

    public TvGatewayDiscoveryManager(Context context) {
        mNsdManager = (NsdManager) context.getSystemService(Context.NSD_SERVICE);
    }

    /** 开始扫描局域网内的Writech网关设备 */
    public void startDiscovery() {
        if (mDiscovering) return;
        mDiscovering = true;
        mGateways.clear();

        mDiscoveryListener = new NsdManager.DiscoveryListener() {
            @Override
            public void onDiscoveryStarted(String serviceType) {
                Log.d(TAG, "NSD discovery started: " + serviceType);
                // 设置超时: 30秒后停止扫描
                mTimeoutHandler.postDelayed(() -> stopDiscovery(), DISCOVERY_TIMEOUT_MS);
            }
        };

        @Override

```

```

        public void onServiceFound(NsdServiceInfo serviceInfo) {
            if (serviceInfo.getServiceName().startsWith(SERVICE_NAME_PREFIX)) {
                Log.d(TAG, "Gateway found: " + serviceInfo.getServiceName());
                resolveService(serviceInfo);
            }
        }

        @Override
        public void onServiceLost(NsdServiceInfo serviceInfo) {
            Log.d(TAG, "Gateway lost: " + serviceInfo.getServiceName());
            // 移除已下线的网关
            mGateways.values().removeIf(gw ->
                gw.serviceName.equals(serviceInfo.getServiceName()));
            notifyGatewayLost(serviceInfo.getServiceName());
        }

        @Override
        public void onDiscoveryStopped(String serviceType) {
            mDiscovering = false;
        }

        @Override
        public void onStartDiscoveryFailed(String serviceType, int errorCode) {
            Log.e(TAG, "Discovery failed: " + errorCode);
            mDiscovering = false;
        }

        @Override
        public void onStopDiscoveryFailed(String serviceType, int errorCode) {
            Log.e(TAG, "Stop discovery failed: " + errorCode);
        }
    };

    mNsdManager.discoverServices(SERVICE_TYPE,
        NsdManager.PROTOCOL_DNS_SD, mDiscoveryListener);
}

/** 解析服务获取IP和端口 */
private void resolveService(NsdServiceInfo serviceInfo) {
    mResolveListener = new NsdManager.ResolveListener() {
        @Override
        public void onResolveFailed(NsdServiceInfo info, int errorCode) {
            Log.e(TAG, "Resolve failed for " + info.getServiceName() + ": " + errorCode);
            // 1秒后重试解析
            mTimeoutHandler.postDelayed(() -> resolveService(serviceInfo), 1000);
        }

        @Override
        public void onServiceResolved(NsdServiceInfo info) {
            String ip = info.getHost().getHostAddress();
            int port = info.getPort();
            String classroomId = extractClassroomId(info);

            GatewayInfo gateway = new GatewayInfo(
                info.getServiceName(), ip, port, classroomId);
            mGateways.put(ip, gateway);

            Log.i(TAG, "Gateway resolved: " + ip + ":" + port
                + " classroom=" + classroomId);
            notifyGatewayDiscovered(gateway);
        }
    };
    mNsdManager.resolveService(serviceInfo, mResolveListener);
}

```

```

/** 从TXT记录中提取教室ID */
private String extractClassroomId(NsdServiceInfo info) {
    Map<String, byte[]> attrs = info.getAttributes();
    byte[] classroomBytes = attrs.get("classroom_id");
    if (classroomBytes != null) {
        return new String(classroomBytes, StandardCharsets.UTF_8);
    }
    return "UNKNOWN";
}

public void stopDiscovery() {
    mTimeoutHandler.removeCallbacksAndMessages(null);
    if (mDiscovering && mDiscoveryListener != null) {
        try {
            mNsdManager.stopServiceDiscovery(mDiscoveryListener);
        } catch (Exception e) {
            Log.e(TAG, "Stop discovery error", e);
        }
    }
    mDiscovering = false;
}

public List<GatewayInfo> getDiscoveredGateways() {
    return new ArrayList<>(mGateways.values());
}
}

```

C.3 WebSocket课堂数据流实现

TV应用通过WebSocket与网关建立长连接，实时接收全班学生的笔迹数据流。

C.3.1 WebSocket连接管理

```

// TvClassroomWebSocketClient.java
public class TvClassroomWebSocketClient {

    private static final int RECONNECT_DELAY_MS = 3000;
    private static final int MAX_RECONNECT_ATTEMPTS = 10;
    private static final int PING_INTERVAL_MS = 30_000;

    private WebSocket mWebSocket;
    private final OkHttpClient mHttpClient;
    private final String mGatewayUrl;
    private final String mClassroomId;
    private final String mDeviceToken;

    private int mReconnectAttempts = 0;
    private volatile boolean mConnected = false;
    private volatile boolean mIntentionalClose = false;

    private final Handler mReconnectHandler = new Handler(Looper.getMainLooper());
    private InkDataListener mInkDataListener;

    public TvClassroomWebSocketClient(String gatewayIp, int port,
        String classroomId, String deviceToken) {
        this.mGatewayUrl = "ws://" + gatewayIp + ":" + port + "/ws/classroom";
        this.mClassroomId = classroomId;
        this.mDeviceToken = deviceToken;
    }
}

```



```

        this.mHttpClient = new OkHttpClient.Builder()
            .pingInterval(PING_INTERVAL_MS, TimeUnit.MILLISECONDS)
            .connectTimeout(10, TimeUnit.SECONDS)
            .readTimeout(0, TimeUnit.MILLISECONDS) // 长连接不超时
            .build();
    }

    public void connect() {
        mIntentionalClose = false;
        mReconnectAttempts = 0;
        doConnect();
    }

    private void doConnect() {
        Request request = new Request.Builder()
            .url(mGatewayUrl)
            .addHeader("X-Classroom-Id", mClassroomId)
            .addHeader("X-Device-Token", mDeviceToken)
            .addHeader("X-Device-Type", "TV_DISPLAY")
            .build();

        mWebSocket = mHttpClient.newWebSocket(request, new WebSocketListener() {
            @Override
            public void onOpen(@NonNull WebSocket webSocket, @NonNull Response response) {
                mConnected = true;
                mReconnectAttempts = 0;
                Log.i(TAG, "WebSocket connected to gateway");
                // 发送注册消息
                sendRegisterMessage();
            }

            @Override
            public void onMessage(@NonNull WebSocket webSocket, @NonNull ByteString bytes) {
                // 解析二进制笔迹数据包
                parseInkPacket(bytes.toByteArray());
            }

            @Override
            public void onMessage(@NonNull WebSocket webSocket, @NonNull String text) {
                // 解析JSON控制消息（开始上课、下课、清屏等）
                parseControlMessage(text);
            }

            @Override
            public void onClosed(@NonNull WebSocket webSocket, int code, @NonNull String reason) {
                mConnected = false;
                Log.w(TAG, "WebSocket closed: " + code + " " + reason);
                if (!mIntentionalClose) {
                    scheduleReconnect();
                }
            }

            @Override
            public void onFailure(@NonNull WebSocket webSocket,
                @NonNull Throwable t, @Nullable Response response) {
                mConnected = false;
                Log.e(TAG, "WebSocket failure", t);
                if (!mIntentionalClose) {
                    scheduleReconnect();
                }
            }
        });
    }
}

```

```

/**
 * 解析二进制笔迹数据包
 * 数据格式: [版本:1B] [学生ID:4B] [x:2B] [y:2B] [压力:1B] [时间戳:4B] [标志:1B] = 15字节/点
 */
private void parseInkPacket(byte[] data) {
    if (data.length < 15) return;
    int offset = 0;
    int version = data[offset++] & 0xFF;
    if (version != 0x01) return; // 不支持的协议版本

    int studentId = ((data[offset] & 0xFF) << 24) | ((data[offset+1] & 0xFF) << 16)
        | ((data[offset+2] & 0xFF) << 8) | (data[offset+3] & 0xFF);
    offset += 4;

    while (offset + 10 <= data.length) {
        float x = (((data[offset] & 0xFF) << 8) | (data[offset+1] & 0xFF)) / 65535.0f;
        float y = (((data[offset+2] & 0xFF) << 8) | (data[offset+3] & 0xFF)) / 65535.0f;
        float pressure = (data[offset+4] & 0xFF) / 255.0f;
        long timestamp = (long)(data[offset+5] & 0xFF) << 24
            | (long)(data[offset+6] & 0xFF) << 16
            | (long)(data[offset+7] & 0xFF) << 8
            | (data[offset+8] & 0xFF);
        boolean isPenUp = (data[offset+9] & 0x01) != 0;
        offset += 10;

        InkPacket packet = new InkPacket(String.valueOf(studentId),
            x, y, pressure, timestamp, isPenUp);
        if (mInkDataListener != null) {
            mInkDataListener.onInkPacketReceived(packet);
        }
    }
}

private void scheduleReconnect() {
    if (mReconnectAttempts >= MAX_RECONNECT_ATTEMPTS) {
        Log.e(TAG, "Max reconnect attempts reached, giving up");
        return;
    }
    long delay = RECONNECT_DELAY_MS * (1L << Math.min(mReconnectAttempts, 4));
    mReconnectAttempts++;
    Log.i(TAG, "Reconnecting in " + delay + "ms (attempt " + mReconnectAttempts + ")");
    mReconnectHandler.postDelayed(this::doConnect, delay);
}

private void sendRegisterMessage() {
    JSONObject msg = new JSONObject();
    try {
        msg.put("type", "REGISTER");
        msg.put("deviceType", "TV_DISPLAY");
        msg.put("classroomId", mClassroomId);
        msg.put("timestamp", System.currentTimeMillis());
    } catch (JSONException e) {
        Log.e(TAG, "JSON error", e);
    }
    mWebSocket.send(msg.toString());
}

public void disconnect() {
    mIntentionalClose = true;
    mReconnectHandler.removeCallbacksAndMessages(null);
    if (mWebSocket != null) {
        mWebSocket.close(1000, "Normal closure");
    }
}

```

```
    public boolean isConnected() { return mConnected; }  
}
```

C.4 TV焦点导航引擎

Android TV应用基于Leanback库实现标准TV焦点导航，同时针对自然写课堂场景进行了自定义优化。

C.4.1 自定义焦点遍历策略

```
// WritechTvFocusHelper.java  
public class WritechTvFocusHelper {  
  
    /**  
     * 为学生列表GridView配置焦点导航  
     * 循环焦点：最后一个→回到第一个，第一个→跳到最后一个  
     */  
    public static void setupStudentGridFocus(RecyclerView recyclerView,  
        int studentCount, int columnsPerRow) {  
        recyclerView.setDescendantFocusability(ViewGroup.FOCUS_AFTER_DESCENDANTS);  
  
        // 配置方向焦点跳转  
        recyclerView.setOnKeyListener((v, keyCode, event) -> {  
            if (event.getAction() != KeyEvent.ACTION_DOWN) return false;  
  
            RecyclerView.LayoutManager lm = recyclerView.getLayoutManager();  
            if (!(lm instanceof GridLayoutManager)) return false;  
  
            GridLayoutManager glm = (GridLayoutManager) lm;  
            int currentPos = getCurrentFocusedPosition(recyclerView);  
  
            switch (keyCode) {  
                case KeyEvent.KEYCODE_DPAD_RIGHT:  
                    // 行末：跳到下一行第一个  
                    if ((currentPos + 1) % columnsPerRow == 0) {  
                        int nextRowFirst = currentPos + 1;  
                        if (nextRowFirst >= studentCount) nextRowFirst = 0;  
                        focusPosition(recyclerView, nextRowFirst);  
                        return true;  
                    }  
                    break;  
                case KeyEvent.KEYCODE_DPAD_LEFT:  
                    // 行首：跳到上一行末  
                    if (currentPos % columnsPerRow == 0) {  
                        int prevRowLast = currentPos - 1;  
                        if (prevRowLast < 0) prevRowLast = studentCount - 1;  
                        focusPosition(recyclerView, prevRowLast);  
                        return true;  
                    }  
                    break;  
                case KeyEvent.KEYCODE_DPAD_DOWN:  
                    // 最后一行：跳回第一行同列  
                    int col = currentPos % columnsPerRow;  
                    int lastRow = (studentCount - 1) / columnsPerRow;  
                    if (currentPos / columnsPerRow == lastRow) {  
                        focusPosition(recyclerView, col);  
                        return true;  
                    }  
                    break;  
            }  
        });  
    }  
}
```

```

        return false;
    });
}

private static int getCurrentFocusedPosition(RecyclerView rv) {
    View focusedChild = rv.getFocusedChild();
    if (focusedChild == null) return 0;
    return rv.getChildAdapterPosition(focusedChild);
}

private static void focusPosition(RecyclerView rv, int position) {
    rv.scrollToPosition(position);
    rv.post(() -> {
        RecyclerView.ViewHolder vh = rv.findViewHolderForAdapterPosition(position);
        if (vh != null) {
            vh.itemView.requestFocus();
        }
    });
}
}
}

```

C.5 书写回放功能实现

TV端支持课后查看任意学生的书写回放，帧动画逐步重现学生书写过程。

C.5.1 回放控制器

```

// TvStrokeReplayController.java
public class TvStrokeReplayController {

    private static final int REPLAY_FPS = 30;
    private static final long FRAME_INTERVAL_MS = 1000 / REPLAY_FPS;

    // 回放速度倍率
    public enum ReplaySpeed {
        SLOW(0.5f), NORMAL(1.0f), FAST(2.0f), VERY_FAST(4.0f);
        public final float multiplier;
        ReplaySpeed(float m) { this.multiplier = m; }
    }

    private final TvInkSurfaceView mSurfaceView;
    private List<InkPoint> mAllPoints; // 全部笔迹点（按时间戳排序）
    private int mCurrentIndex = 0;
    private long mStartRealTime;
    private long mStartStrokeTime;
    private ReplaySpeed mSpeed = ReplaySpeed.NORMAL;
    private volatile boolean mPlaying = false;
    private volatile boolean mPaused = false;

    private final Handler mReplayHandler = new Handler(Looper.getMainLooper());

    public TvStrokeReplayController(TvInkSurfaceView surfaceView) {
        this.mSurfaceView = surfaceView;
    }

    /**
     * 加载回放数据并开始播放
     * @param points 已按时间戳排序的笔迹点列表
     */
    public void startReplay(List<InkPoint> points) {

```

```

        mAllPoints = new ArrayList<>(points);
        mCurrentIndex = 0;
        mSurfaceView.clearAllInk();

        if (mAllPoints.isEmpty()) return;

        mStartRealTime = SystemClock.elapsedRealtime();
        mStartStrokeTime = mAllPoints.get(0).timestamp;
        mPlaying = true;
        mPaused = false;
        scheduleNextFrame();
    }

    private void scheduleNextFrame() {
        if (!mPlaying || mPaused) return;
        mReplayHandler.postDelayed(this::advanceFrame, FRAME_INTERVAL_MS);
    }

    private void advanceFrame() {
        if (!mPlaying || mPaused || mCurrentIndex >= mAllPoints.size()) {
            if (mCurrentIndex >= mAllPoints.size()) {
                onReplayFinished();
            }
            return;
        }

        long realElapsed = SystemClock.elapsedRealtime() - mStartRealTime;
        long strokeElapsed = (long)(realElapsed * mSpeed.multiplier);
        long targetStrokeTime = mStartStrokeTime + strokeElapsed;

        // 推送所有时间戳 <= targetStrokeTime 的点
        while (mCurrentIndex < mAllPoints.size()) {
            InkPoint pt = mAllPoints.get(mCurrentIndex);
            if (pt.timestamp > targetStrokeTime) break;
            mSurfaceView.pushInkPacket(new InkPacket(
                pt.studentId, pt.x, pt.y, pt.pressure,
                pt.timestamp, pt.isPenUp));
            mCurrentIndex++;
        }
        scheduleNextFrame();
    }

    public void pause() {
        mPaused = true;
        mReplayHandler.removeCallbacksAndMessages(null);
    }

    public void resume() {
        if (!mPaused) return;
        // 重新校准时间基准
        if (mCurrentIndex < mAllPoints.size()) {
            mStartRealTime = SystemClock.elapsedRealtime();
            mStartStrokeTime = mAllPoints.get(mCurrentIndex).timestamp;
        }
        mPaused = false;
        scheduleNextFrame();
    }

    public void setSpeed(ReplaySpeed speed) {
        this.mSpeed = speed;
    }

    public void seekTo(float progress) {
        // progress: [0.0, 1.0]
    }

```

```

        int targetIndex = (int)(progress * mAllPoints.size());
        targetIndex = Math.max(0, Math.min(targetIndex, mAllPoints.size() - 1));

        // 清屏并重新绘制到目标位置
        mSurfaceView.clearAllInk();
        for (int i = 0; i <= targetIndex; i++) {
            InkPoint pt = mAllPoints.get(i);
            mSurfaceView.pushInkPacket(new InkPacket(
                pt.studentId, pt.x, pt.y, pt.pressure,
                pt.timestamp, pt.isPenUp));
        }
        mCurrentIndex = targetIndex;

        if (!mPaused && mCurrentIndex < mAllPoints.size()) {
            mStartRealTime = SystemClock.elapsedRealtime();
            mStartStrokeTime = mAllPoints.get(mCurrentIndex).timestamp;
        }
    }

    public void stop() {
        mPlaying = false;
        mPaused = false;
        mReplayHandler.removeCallbacksAndMessages(null);
        mSurfaceView.clearAllInk();
    }

    private void onReplayFinished() {
        mPlaying = false;
        Log.i(TAG, "Replay finished, total points: " + mAllPoints.size());
    }

    public boolean isPlaying() { return mPlaying && !mPaused; }
    public int getProgress() {
        if (mAllPoints == null || mAllPoints.isEmpty()) return 0;
        return (int)(100.0f * mCurrentIndex / mAllPoints.size());
    }
}

```

C.6 课件展示模块

TV端支持展示PPT/PDF课件，与教师端投影内容同步。

C.6.1 课件展示Activity

```

// TvCoursewareActivity.java
public class TvCoursewareActivity extends Activity {

    private static final String EXTRA_COURSEWARE_URL = "courseware_url";
    private static final String EXTRA_COURSEWARE_TYPE = "courseware_type"; // PDF/IMAGE

    private ViewPager2 mSlidePager;
    private TvCoursewareAdapter mAdapter;
    private List<Bitmap> mSlides = new ArrayList<>();

    private int mCurrentSlide = 0;
    private int mTotalSlides = 0;

    // 监听教师端翻页指令
    private WebSocketCommandListener mCommandListener;
}

```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_tv_courseware);

    // 全屏无状态栏
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN);
    getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

    mSlidePager = findViewById(R.id.slide_pager);
    mAdapter = new TvCoursewareAdapter(mSlides);
    mSlidePager.setAdapter(mAdapter);

    String coursewareUrl = getIntent().getStringExtra(EXTRA_COURSEWARE_URL);
    String type = getIntent().getStringExtra(EXTRA_COURSEWARE_TYPE);

    if ("PDF".equals(type)) {
        loadPdfCourseware(coursewareUrl);
    } else {
        loadImageCourseware(coursewareUrl);
    }

    setupRemoteControl();
    setupCommandListener();
}

/**
 * 异步加载PDF课件，使用PdfRenderer逐页渲染为Bitmap
 */
private void loadPdfCourseware(String url) {
    new Thread(() -> {
        try {
            // 下载PDF到本地缓存
            File pdfFile = downloadToCache(url);
            ParcelFileDescriptor pfd = ParcelFileDescriptor.open(
                pdfFile, ParcelFileDescriptor.MODE_READ_ONLY);
            PdfRenderer renderer = new PdfRenderer(pfd);
            int pageCount = renderer.getPageCount();

            for (int i = 0; i < pageCount; i++) {
                PdfRenderer.Page page = renderer.openPage(i);
                int width = 1920; // TV 1080p宽度
                int height = (int)(1.0f * page.getHeight() / page.getWidth() * width);
                Bitmap bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.RGB_565);
                page.render(bitmap, null, null, PdfRenderer.Page.RENDER_MODE_FOR_DISPLAY);
                page.close();

                final int index = i;
                runOnUiThread(() -> {
                    mSlides.add(bitmap);
                    mAdapter.notifyItemInserted(mSlides.size() - 1);
                    if (index == 0) {
                        mTotalSlides = pageCount;
                        updateSlideCounter(1, pageCount);
                    }
                });
            }
            renderer.close();
        } catch (Exception e) {
            Log.e(TAG, "PDF load failed", e);
            runOnUiThread(() -> showErrorHint("课件加载失败，请检查网络连接"));
        }
    }).start();
}

```

```

/**
 * 响应教师端翻页控制指令
 */
private void setupCommandListener() {
    mCommandListener = new WebSocketCommandListener() {
        @Override
        public void onCommand(String command, JSONObject payload) {
            switch (command) {
                case "SLIDE_NEXT":
                    runOnUiThread(() -> nextSlide());
                    break;
                case "SLIDE_PREV":
                    runOnUiThread(() -> prevSlide());
                    break;
                case "SLIDE_GOTO":
                    int page = payload.optInt("page", 1);
                    runOnUiThread(() -> gotoSlide(page - 1));
                    break;
                case "ANNOTATION_SYNC":
                    // 同步教师标注（叠加到当前幻灯片上层）
                    runOnUiThread(() -> syncAnnotation(payload));
                    break;
            }
        }
    };
}

private void nextSlide() {
    if (mCurrentSlide < mTotalSlides - 1) {
        mCurrentSlide++;
        mSlidePager.setCurrentItem(mCurrentSlide, true);
        updateSlideCounter(mCurrentSlide + 1, mTotalSlides);
    }
}

private void prevSlide() {
    if (mCurrentSlide > 0) {
        mCurrentSlide--;
        mSlidePager.setCurrentItem(mCurrentSlide, true);
        updateSlideCounter(mCurrentSlide + 1, mTotalSlides);
    }
}

private void gotoSlide(int index) {
    if (index >= 0 && index < mTotalSlides) {
        mCurrentSlide = index;
        mSlidePager.setCurrentItem(index, false);
        updateSlideCounter(index + 1, mTotalSlides);
    }
}

private void updateSlideCounter(int current, int total) {
    TextView counter = findViewById(R.id.slide_counter);
    counter.setText(current + " / " + total);
}

/** 遥控器左右键翻页 */
private void setupRemoteControl() {
    mSlidePager.setOnKeyListener((v, keyCode, event) -> {
        if (event.getAction() != KeyEvent.ACTION_DOWN) return false;
        switch (keyCode) {
            case KeyEvent.KEYCODE_DPAD_RIGHT:
            case KeyEvent.KEYCODE_MEDIA_FAST_FORWARD:

```



```

        nextSlide(); return true;
        case KeyEvent.KEYCODE_DPAD_LEFT:
        case KeyEvent.KEYCODE_MEDIA_REWIND:
            prevSlide(); return true;
    }
    return false;
});
}
}

```

C.7 TV主界面布局与Leanback框架集成

```

// TvMainFragment.java - 使用Leanback BrowseSupportFragment
public class TvMainFragment extends BrowseSupportFragment {

    private static final String HEADER_CLASSROOM = "课堂";
    private static final String HEADER_HOMEWORK = "作业";
    private static final String HEADER_REPLAY = "回放";
    private static final String HEADER_REPORT = "报告";

    private ArrayObjectAdapter mRowsAdapter;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setupUIElements();
        loadRows();
    }

    private void setupUIElements() {
        setTitle("自然写互动课堂");
        setHeadersState(HEADERS_ENABLED);
        setHeadersTransitionOnBackEnabled(true);
        setBrandColor(getResources().getColor(R.color.writech_primary));
        setSearchAffordanceColor(getResources().getColor(R.color.writech_accent));
    }

    private void loadRows() {
        mRowsAdapter = new ArrayObjectAdapter(new ListRowPresenter());

        // 课堂功能行
        HeaderItem classroomHeader = new HeaderItem(0, HEADER_CLASSROOM);
        ArrayObjectAdapter classroomAdapter = new ArrayObjectAdapter(new CardPresenter());
        classroomAdapter.add(new CardItem("进入课堂", R.drawable.ic_classroom,
CardItem.TYPE_CLASSROOM));
        classroomAdapter.add(new CardItem("全班展示", R.drawable.ic_all_students,
CardItem.TYPE_ALL_DISPLAY));
        classroomAdapter.add(new CardItem("答题收集", R.drawable.ic_answer,
CardItem.TYPE_ANSWER_COLLECT));
        classroomAdapter.add(new CardItem("白板模式", R.drawable.ic_whiteboard,
CardItem.TYPE_WHITEBOARD));
        mRowsAdapter.add(new ListRow(classroomHeader, classroomAdapter));

        // 作业功能行
        HeaderItem homeworkHeader = new HeaderItem(1, HEADER_HOMEWORK);
        ArrayObjectAdapter homeworkAdapter = new ArrayObjectAdapter(new CardPresenter());
        homeworkAdapter.add(new CardItem("批改作业", R.drawable.ic_grade, CardItem.TYPE_GRADE));
        homeworkAdapter.add(new CardItem("作业统计", R.drawable.ic_stats, CardItem.TYPE_STATS));
        mRowsAdapter.add(new ListRow(homeworkHeader, homeworkAdapter));

        // 回放功能行
        HeaderItem replayHeader = new HeaderItem(2, HEADER_REPLAY);
    }
}

```

```

        ArrayObjectAdapter replayAdapter = new ArrayObjectAdapter(new CardPresenter());
        replayAdapter.add(new CardItem("课堂回放", R.drawable.ic_replay, CardItem.TYPE_REPLAY));
        replayAdapter.add(new CardItem("学生对比", R.drawable.ic_compare, CardItem.TYPE_COMPARE));
        mRowsAdapter.add(new ListRow(replayHeader, replayAdapter));

        setAdapter(mRowsAdapter);
    }

    @Override
    public void onItemClickClickedListener() {
        setOnItemClickListener((itemViewHolder, item, rowViewHolder, row) -> {
            if (item instanceof CardItem) {
                CardItem card = (CardItem) item;
                handleCardClick(card);
            }
        });
    }

    private void handleCardClick(CardItem card) {
        Intent intent;
        switch (card.getType()) {
            case CardItem.TYPE_CLASSROOM:
                intent = new Intent(getActivity(), TvClassroomActivity.class);
                startActivity(intent);
                break;
            case CardItem.TYPE_REPLAY:
                intent = new Intent(getActivity(), TvReplayActivity.class);
                startActivity(intent);
                break;
            default:
                Toast.makeText(getActivity(), "功能开发中", Toast.LENGTH_SHORT).show();
        }
    }
}

```

附录D 完整操作手册

D.1 安装与初始配置

D.1.1 安装步骤

1. 将自然写TV应用APK文件通过U盘拷贝至电视USB接口，或通过学校内网推送安装。
2. 打开电视"应用管理"→"安装外部应用"，选择APK文件。
3. 安装完成后，应用图标出现在电视应用列表中（图标：蓝色书写笔图案）。
4. 首次启动时，系统提示申请以下权限：
5. 网络访问权限（必需）
6. 局域网服务发现权限（必需）
7. 存储读写权限（用于课件缓存）
8. 点击"全部允许"完成权限授权。

D.1.2 网络配置要求

配置项	要求
网络类型	WiFi或有线以太网（推荐有线）
网络带宽	≥ 10Mbps（保障全班笔迹实时传输）
网络类型	与网关设备处于同一局域网
防火墙	开放UDP 5353（mDNS）、TCP 8765（WebSocket）
IP分配	建议电视设备使用静态IP，避免DHCP变更影响连接稳定性

D.1.3 首次登录

1. 打开自然写TV应用，进入账号登录页面。
2. 显示两种登录方式：
3. **账号登录**：输入学校管理员账号和密码
4. **扫码登录**：手机打开自然写APP扫描TV屏幕二维码授权
5. 登录成功后，系统自动扫描局域网内的网关设备（约5–15秒）。
6. 发现网关后，TV应用自动绑定当前教室，进入主界面。

D.2 课堂功能操作

D.2.1 进入课堂模式

1. 在主界面选择"课堂"栏目，遥控器确认键进入。
2. 选择"进入课堂"，TV屏幕切换为班级笔迹显示界面。
3. 界面布局：
4. 顶部状态栏：教室名称、在线学生数、当前时间
5. 主区域：4×8网格显示32个学生的实时笔迹区域（每格显示学生姓名）
6. 底部工具栏：切换视图/全屏展示/清屏/课件/退出

D.2.2 全班笔迹展示操作

操作	遥控器按键	说明
切换至单生全屏	选中格子 → 确认键	放大显示选中学生笔迹
返回全班视图	返回键	从单生全屏返回网格视图
标记优秀作品	选中格子 → 菜单键 → 标记	为该学生笔迹添加星标
全班清屏	工具栏选"清屏" → 确认	清除所有学生当前笔迹
单生清屏	选中格子 → 菜单键 → 清屏	仅清除选中学生笔迹

D.2.3 答题收集操作

1. 在课堂界面底部工具栏选择"答题收集"。
2. 设置答题参数：

3. 答题时限（1-10分钟，默认3分钟）
4. 是否允许修改（倒计时结束前可重复书写）
5. 确认后TV屏幕显示倒计时，所有学生Pad/智能笔进入答题锁定模式。
6. 答题时限到达后，TV屏幕自动展示全班答题结果网格视图。
7. 教师可通过遥控器浏览各学生答案，按"菜单键"可操作：
8. 标记正确/错误
9. 展示到投影（全屏单生）
10. 添加批注（手写板或遥控键盘输入）

D.2.4 白板模式

1. 选择"白板模式"后，TV进入纯白背景白板界面。
2. 此模式支持通过网关接收教师智能笔书写内容实时展示。
3. 工具栏支持：
4. 画笔颜色（8种颜色）
5. 画笔粗细（细/中/粗/超粗）
6. 橡皮擦（选中后遥控器确认键点击区域）
7. 撤销/重做（遥控器媒体键控制）
8. 清屏
9. 截图保存（保存到本地存储用于分享）

D.3 回放功能操作

D.3.1 查看课堂回放

1. 主界面选择"回放"栏目，确认进入。
2. 显示历史课堂列表（按日期排序），选择要回放的课堂。
3. 进入回放界面：
4. 顶部：回放进度条（可左右键拖动）
5. 主区域：全班笔迹动态重现
6. 右侧：学生列表（可单选/全选）
7. 底部控制栏：播放/暂停/速度/截图

D.3.2 回放控制操作

操作	遥控器按键	说明
播放/暂停	确认键 或 播放/暂停键	切换播放状态
快进	快进键 / 右键长按	速度×2，再按×4
快退	快退键 / 左键长按	速度×0.5
跳到开头	左键×3快按	从头开始回放
跳到结尾	右键×3快按	快速预览最终结果
进度跳转	上键选中进度条 → 左右键	拖动进度条跳转

D.4 报告查看

D.4.1 学情报告展示

1. 主界面选择"报告"栏目，确认进入。
2. 支持查看：
3. 班级整体报告（正确率分布图、作业完成率等）
4. 单生报告（选择学生姓名后展示）
5. TV端报告为只读展示模式，大字体适配远距离观看。
6. 可通过遥控器上下键翻页浏览报告内容。

D.5 常见问题处理

现象	可能原因	处理方法
启动后无法发现网关	网关未上电 或 WiFi网络不同	检查网关LED指示灯，确认TV与网关同一WiFi
学生笔迹显示卡顿	网络带宽不足	切换至有线以太网，或减少同时连接的学生设备数
课件加载失败	云端存储连接超时	检查网络，或使用U盘本地课件
回放数据缺失	课堂数据未上传完成	等待数据同步完成（状态栏显示同步进度）
遥控器无响应	焦点丢失	按HOME键回到主界面重新进入
画面撕裂	GPU渲染性能不足	在"设置→显示"中降低渲染分辨率至1080p

附录E 源代码详细对应关系

E.1 完整源代码文件清单

源文件	路径	功能说明
TvMainActivity.java	app/src/main/java/.../tv/TvMainActivity.java	TV主界面Activity, Leanback入口
TvMainFragment.java	.../tv/TvMainFragment.java	BrowseSupportFragment, 主导航
TvClassroomActivity.java	.../classroom/TvClassroomActivity.java	课堂模式主Activity
TvInkSurfaceView.java	.../view/TvInkSurfaceView.java	双缓冲笔迹渲染 SurfaceView
StudentInkState.java	.../model/StudentInkState.java	单学生笔迹状态管理
TvClassroomWebSocketClient.java	.../network/TvClassroomWebSocketClient.java	WebSocket课堂连接
TvGatewayDiscoveryManager.java	.../network/TvGatewayDiscoveryManager.java	mDNS网关发现

源文件	路径	功能说明
TvStrokeReplayController.java	.../replay/TvStrokeReplayController.java	书写回放控制器
TvCoursewareActivity.java	.../courseware/TvCoursewareActivity.java	课件展示Activity
WritechTvFocusHelper.java	.../util/WritechTvFocusHelper.java	焦点导航工具类
CardPresenter.java	.../presenter/CardPresenter.java	Leanback卡片Presenter
StudentColorPalette.java	.../util/StudentColorPalette.java	学生笔迹颜色分配
InkPacket.java	.../model/InkPacket.java	笔迹数据包模型
GatewayInfo.java	.../model/GatewayInfo.java	网关设备信息模型
RenderConfig.java	.../config/RenderConfig.java	渲染配置常量
TvAnswerCollectFragment.java	.../answer/TvAnswerCollectFragment.java	答题收集Fragment
TvWhiteboardActivity.java	.../whiteboard/TvWhiteboardActivity.java	白板模式Activity
TvReportActivity.java	.../report/TvReportActivity.java	学情报告展示Activity
TvSettingsActivity.java	.../settings/TvSettingsActivity.java	设置界面Activity

E.2 核心功能函数说明

函数名	所属类	说明
surfaceCreated()	TvInkSurfaceView	SurfaceView创建时初始化双缓冲
processInkQueue()	RenderThread	消费笔迹队列，更新Path
drawFrame()	RenderThread	绘制当前帧所有学生笔迹
swapBuffers()	RenderThread	前后缓冲区交换并提交
addPoint()	StudentInkState	添加笔迹点（贝塞尔平滑）
startDiscovery()	TvGatewayDiscoveryManager	启动mDNS网关扫描
resolveService()	TvGatewayDiscoveryManager	解析服务获取IP/端口
connect()	TvClassroomWebSocketClient	建立WebSocket长连接
parseInkPacket()	TvClassroomWebSocketClient	解析二进制笔迹数据包
scheduleReconnect()	TvClassroomWebSocketClient	指数退避重连调度
startReplay()	TvStrokeReplayController	启动书写回放
advanceFrame()	TvStrokeReplayController	推进回放帧
seekTo()	TvStrokeReplayController	进度条跳转
setupStudentGridFocus()	WritechTvFocusHelper	配置网格焦点循环导航

函数名	所属类	说明
loadPdfCourseware()	TvCoursewareActivity	异步加载PDF课件

E.3 数据模型说明

```
// InkPacket.java - 笔迹数据包
public class InkPacket {
    public final String studentId; // 学生ID (对应学生座位编号)
    public final float x;          // 归一化x坐标 [0.0, 1.0]
    public final float y;          // 归一化y坐标 [0.0, 1.0]
    public final float pressure;   // 压力值 [0.0, 1.0]
    public final long timestamp;   // 毫秒时间戳
    public final boolean isPenUp;  // true=抬笔 (笔画结束)

    public InkPacket(String studentId, float x, float y,
                     float pressure, long timestamp, boolean isPenUp) {
        this.studentId = studentId;
        this.x = x;
        this.y = y;
        this.pressure = pressure;
        this.timestamp = timestamp;
        this.isPenUp = isPenUp;
    }
}

// GatewayInfo.java - 网关设备信息
public class GatewayInfo {
    public final String serviceName; // mDNS服务名称
    public final String ipAddress;   // 网关IP地址
    public final int port;           // WebSocket端口
    public final String classroomId; // 教室ID (从TXT记录获取)
    public long lastSeenTime;        // 最近一次发现时间

    public GatewayInfo(String serviceName, String ipAddress,
                      int port, String classroomId) {
        this.serviceName = serviceName;
        this.ipAddress = ipAddress;
        this.port = port;
        this.classroomId = classroomId;
        this.lastSeenTime = System.currentTimeMillis();
    }
}
```

E.4 AndroidManifest关键配置

```
<!-- AndroidManifest.xml (TV相关关键配置) -->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.writech.tv">

    <!-- TV应用必需权限 -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
        android:maxSdkVersion="28" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

```
<!-- 声明为TV应用 -->
<uses-feature
    android:name="android.hardware.touchscreen"
    android:required="false" />
<uses-feature
    android:name="android.software.leanback"
    android:required="true" />

<application
    android:name=".WritechTvApplication"
    android:label="@string/app_name"
    android:icon="@drawable/ic_launcher"
    android:banner="@drawable/ic_banner"
    android:theme="@style/Theme.Leanback">

    <!-- TV主入口 -->
    <activity
        android:name=".TvMainActivity"
        android:exported="true"
        android:screenOrientation="landscape"
        android:configChanges="keyboard|keyboardHidden|navigation">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LEANBACK_LAUNCHER" />
        </intent-filter>
    </activity>

    <!-- 课堂Activity -->
    <activity
        android:name=".classroom.TvClassroomActivity"
        android:exported="false"
        android:screenOrientation="landscape"
        android:launchMode="singleTask"
        android:keepScreenOn="true" />

    <!-- 白板Activity -->
    <activity
        android:name=".whiteboard.TvWhiteboardActivity"
        android:exported="false"
        android:screenOrientation="landscape"
        android:launchMode="singleTask" />

</application>
</manifest>
```

附录F 性能测试数据

F.1 渲染性能指标

测试场景	平均帧率	P99延迟	内存占用
空载（无学生连接）	60fps	2ms	128MB
10名学生同时书写	60fps	8ms	198MB
20名学生同时书写	58fps	15ms	267MB
32名学生同时书写	55fps	28ms	356MB

测试场景	平均帧率	P99延迟	内存占用
32名学生+课件展示	50fps	35ms	412MB

测试设备：Xiaomi Mi Box 4K (Amlogic S905X4, 2GB RAM), Android TV 11

F.2 网络延迟指标

测试场景	端到端延迟 (笔点→TV显示)	丢包率
有线千兆局域网	< 20ms	0%
WiFi 5GHz局域网	< 35ms	< 0.1%
WiFi 2.4GHz局域网	< 60ms	< 0.5%
跨WiFi路由器	< 80ms	< 1%

F.3 稳定性测试

测试项目	测试时长	结果
连续课堂运行	8小时	无崩溃, 内存无泄漏
网络中断重连	100次	100%成功重连, 平均重连时间3.2秒
压力测试 (32生高频书写)	30分钟	帧率保持≥50fps, 无数据丢失

文档编制：深圳自然写科技有限公司 Android TV研发团队

文档版本：V1.0 (附录更新)

最后更新：2026年2月14日

版权所有 © 2026 深圳自然写科技有限公司

附录G 性能与兼容性

G.1 兼容设备清单

品牌	型号	芯片	Android TV版本	测试结果
小米	Mi Box 4K	Amlogic S905X4	Android TV 11	完全兼容
索尼	X85J	MT5896	Android TV 10	完全兼容
飞利浦	PUS8536	MT9950	Android TV 11	完全兼容
创维	E5 Pro	Mstar 6A928	Android TV 9	基本兼容
海信	U7H	MT9620	Android TV 11	完全兼容

G.2 Android TV与手机版差异对比

功能	TV版	手机/Pad版
交互方式	遥控器D-Pad导航	触摸屏手势
布局设计	大字体、高对比度、焦点突出	正常字体、密度信息
笔迹输入	WebSocket接收全班笔迹（只展示）	BLE直接接收本机智能笔
白板书写	通过网关接收教师智能笔输入	直接触摸或BLE智能笔
课件展示	PDF/PPT投影+同步翻页	学生视角课件浏览
数据上传	仅展示，不上传	直接上传笔迹和作业

G.3 TV应用权限说明

权限	用途	是否必需
INTERNET	网络连接、WebSocket通信	必需
ACCESS_NETWORK_STATE	检测网络状态	必需
ACCESS_WIFI_STATE	获取WiFi信息	必需
CHANGE_NETWORK_STATE	切换网络配置	可选
READ_EXTERNAL_STORAGE	读取U盘课件	可选
WRITE_EXTERNAL_STORAGE	保存课堂截图	可选
RECORD_AUDIO	课堂音频监听（可选功能）	可选
RECEIVE_BOOT_COMPLETED	开机自动启动	可选

本文档版权归深圳自然写科技有限公司所有，仅用于软件著作权登记鉴别，请勿用于其他商业用途。

附录H 源代码目录结构

```
app/src/main/java/com/writetech/tv/
├── TvApplication.java           # Application初始化
├── TvMainActivity.java          # TV主入口Activity
├── TvMainFragment.java         # Leanback BrowseSupportFragment
├── classroom/
│   ├── TvClassroomActivity.java # 课堂主Activity
│   ├── TvInkSurfaceView.java    # 双缓冲笔迹渲染SurfaceView
│   ├── StudentInkState.java     # 单学生笔迹状态
│   ├── TvAnswerCollectFragment.java # 答题收集Fragment
│   └── TvWhiteboardActivity.java # 白板模式Activity
├── network/
│   ├── TvClassroomWebSocketClient.java # WebSocket课堂连接
│   └── TvGatewayDiscoveryManager.java  # mDNS网关发现
```

```

├─ courseware/
│   └─ TvCoursewareActivity.java # 课件展示Activity
├─ replay/
│   └─ TvStrokeReplayController.java # 书写回放控制器
├─ report/
│   └─ TvReportActivity.java # 学情报告展示
├─ presenter/
│   └─ CardPresenter.java # Leanback卡片Presenter
├─ model/
│   ├── InkPacket.java # 笔迹数据包模型
│   ├── GatewayInfo.java # 网关信息模型
│   └─ CardItem.java # 主界面卡片数据模型
├─ util/
│   ├── WritetechTvFocusHelper.java # TV焦点导航工具
│   └─ StudentColorPalette.java # 学生颜色分配工具
└─ settings/
    └─ TvSettingsActivity.java # 设置页面

app/src/main/res/
├─ layout/
│   ├── activity_tv_classroom.xml # 课堂界面布局（全屏）
│   ├── activity_tv_courseware.xml # 课件展示布局
│   └─ fragment_tv_main.xml # 主界面布局
├─ drawable/
│   ├── ic_launcher.xml # 应用图标
│   └─ ic_banner.xml # TV端横幅（320×180dp）
└─ values/
    ├── strings.xml # 字符串资源
    └─ themes.xml # Leanback主题配置

```

H.1 Gradle依赖配置

```

// build.gradle (app)
dependencies {
    implementation 'androidx.leanback:leanback:1.2.0'
    implementation 'com.squareup.okhttp3:okhttp:4.12.0'
    implementation 'com.squareup.okhttp3:okhttp-ws:4.12.0'
    implementation 'com.google.code.gson:gson:2.10.1'
    implementation 'com.github.bumptech.glide:glide:4.16.0'
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.7.3'
}

```

本文档版权归深圳自然写科技有限公司所有，技术细节仅用于软件著作权登记鉴别。

附录F 补充技术规格

F.1 Focus焦点导航引擎

Android TV应用不依赖触摸屏，完全通过遥控器D-Pad操控，焦点管理是核心技术：

```

// FocusNavigationManager.kt
class FocusNavigationManager(private val activity: AppCompatActivity) {

    // 自定义焦点遍历顺序

```

```

fun setupStudentGridFocus(gridLayout: GridLayout) {
    val children = (0 until gridLayout.childCount).map {
        gridLayout.getChildAt(it)
    }

    val cols = gridLayout.columnCount
    children.forEachIndexed { index, view ->
        val row = index / cols
        val col = index % cols

        // 设置四个方向的焦点目标
        view.nextFocusUpId = children.getOrNull(index - cols)?.id ?: View.NO_ID
        view.nextFocusDownId = children.getOrNull(index + cols)?.id ?: View.NO_ID
        view.nextFocusLeftId = if (col > 0) children[index - 1].id else View.NO_ID
        view.nextFocusRightId = if (col < cols - 1) children[index + 1].id else View.NO_ID

        view.isFocusable = true
        view.setOnFocusChangeListener { v, hasFocus ->
            v.animate().scaleX(if (hasFocus) 1.1f else 1.0f)
                .scaleY(if (hasFocus) 1.1f else 1.0f)
                .setDuration(150).start()
        }
    }
}

// 记住并恢复焦点位置
fun saveFocusState(): Bundle {
    val focused = activity.currentFocus
    return Bundle().apply {
        putInt("focused_id", focused?.id ?: View.NO_ID)
    }
}

fun restoreFocusState(state: Bundle) {
    val id = state.getInt("focused_id")
    if (id != View.NO_ID) {
        activity.findViewById<View>(id)?.requestFocus()
    }
}
}

```

F.2 4K分辨率适配

```

// DisplayAdapter.kt
object DisplayAdapter {
    fun getOptimalLayoutConfig(context: Context): LayoutConfig {
        val dm = context.resources.displayMetrics
        val widthPx = dm.widthPixels
        val heightPx = dm.heightPixels

        return when {
            widthPx >= 3840 -> LayoutConfig( // 4K UHD
                studentCols = 8,
                studentRows = 5,
                fontSize = 28f,
                inkScaleFactor = 4.0f,
                boardPadding = 48
            )
            widthPx >= 1920 -> LayoutConfig( // 1080p Full HD
                studentCols = 6,
                studentRows = 4,
                fontSize = 22f,

```

```

        inkScaleFactor = 2.0f,
        boardPadding = 32
    )
    else -> LayoutConfig( // 720p HD
        studentCols = 4,
        studentRows = 3,
        fontSize = 18f,
        inkScaleFactor = 1.5f,
        boardPadding = 24
    )
}
}

data class LayoutConfig(
    val studentCols: Int,
    val studentRows: Int,
    val fontSize: Float,
    val inkScaleFactor: Float,
    val boardPadding: Int
)

```

F.3 课堂互动答题统计

```

// AnswerStatisticsEngine.kt
class AnswerStatisticsEngine {
    data class AnswerStats(
        val optionCounts: Map<String, Int>, // 各选项人数
        val correctRate: Float, // 正确率
        val avgSubmitTime: Long, // 平均提交耗时 (ms)
        val totalStudents: Int,
        val submittedCount: Int
    )

    fun compute(answers: List<StudentAnswer>, correctAnswer: String): AnswerStats {
        val counts = answers.groupingBy { it.answer }.eachCount()
        val correct = counts[correctAnswer] ?: 0
        val avgTime = if (answers.isEmpty()) 0L
            else answers.sumOf { it.submitTime - it.questionStartTime } / answers.size

        return AnswerStats(
            optionCounts = counts,
            correctRate = if (answers.isEmpty()) 0f
                else correct.toFloat() / answers.size,
            avgSubmitTime = avgTime,
            totalStudents = answers.size,
            submittedCount = answers.count { it.submitted }
        )
    }

    // 生成答题分布柱状图数据 (用于Canvas绘制)
    fun buildChartData(stats: AnswerStats): List<BarEntry> {
        val options = listOf("A", "B", "C", "D")
        return options.mapIndexed { i, opt ->
            BarEntry(i.toFloat(), (stats.optionCounts[opt] ?: 0).toFloat())
        }
    }
}

```

本文档版权归深圳自然写科技有限公司所有，技术细节仅用于软件著作权登记鉴别。