# Dr33MTV Roku Channel Development Guide

## Platform Overview

Dr33MTV is a revolutionary global music streaming channel for Roku devices, featuring AI-powered content discovery, high-quality streaming, and integration with the 1DreamUnited ecosystem.

## Development Environment Setup

### Prerequisites

- **Roku Device** (Roku 2 or newer recommended)
- **Network Connection** (Roku and development machine on same network)
- **Roku Developer Account** (free registration at developer.roku.com)
- **Text Editor** or **IDE** (VS Code recommended with BrightScript extension)

### Roku Device Setup

1. **Enable Developer Mode**:
   - Go to Settings > System > Advanced system settings > Developer options
   - Enable "Developer mode"
   - Set a developer password (remember this!)
   - Note the device IP address

2. **Network Configuration**:
   - Ensure Roku device is connected to your local network
   - Note the IP address (Settings > Network > About)

## Getting Started

### 1. Project Structure

```
roku/
├── components/              # UI components
│   ├── HomeScene.brs       # Main scene logic
│   ├── HomeScene.xml       # Main scene layout
│   ├── ContentGrid.brs     # Content grid logic
│   ├── ContentGrid.xml     # Content grid layout
│   ├── VideoPlayer.brs     # Video player logic
│   └── VideoPlayer.xml     # Video player layout
├── source/                 # BrightScript source files
│   ├── main.brs           # Application entry point
│   └── ContentApi.brs     # API integration
├── images/                # Channel assets
│   ├── Icon_HD.png        # HD channel icon
│   ├── Icon_SD.png        # SD channel icon
│   ├── splash_hd.jpg      # HD splash screen
│   └── splash_sd.jpg      # SD splash screen
├── locale/                # Localization files
├── manifest               # Channel manifest
└── deploy.sh             # Deployment script
```

### 2. Channel Manifest

```
# manifest
title=Dr33MTV
subtitle=Revolutionary Global Music Streaming
major_version=1
minor_version=0
build_version=00001

mm_icon_focus_hd=pkg:/images/Icon_HD.png
mm_icon_focus_sd=pkg:/images/Icon_SD.png
splash_screen_hd=pkg:/images/splash_hd.jpg
splash_screen_sd=pkg:/images/splash_sd.jpg
splash_color=#000000
splash_min_time=1000

ui_resolutions=hd
rsg_version=1.2
```

## Development Process

### 1. Local Development

```
# Navigate to Roku project
cd roku/

# Package the channel
zip -r Dr33MTV_Channel.zip . -x "*.git*" "*.DS_Store" "deploy.sh" "README.md"
```

### 2. Deployment to Roku Device

```
# Set environment variables
export ROKU_DEV_TARGET=192.168.1.XXX  # Your Roku IP
export ROKU_DEV_PASSWORD=your_dev_password

# Deploy using the script
./deploy.sh

# Or deploy manually via web interface
# Open http://ROKU_IP in browser
# Login with username "rokudev" and your developer password
# Upload the Dr33MTV_Channel.zip file
```

### 3. Testing and Debugging

- **Remote Control**: Use Roku remote or mobile app
- **Debug Console**: Access via telnet to port 8085
- **Performance**: Monitor via telnet to port 8080

## Key Components

### Main Scene (HomeScene)

```
' HomeScene.brs
function init()
    m.top.backgroundURI = "pkg:/images/background.jpg"
    m.contentGrid = m.top.findNode("contentGrid")
    m.contentTask = createObject("roSGNode", "ContentTask")
    m.contentTask.observeField("content", "onContentLoaded")
    m.contentTask.control = "RUN"
end function

function onContentLoaded()
    m.contentGrid.content = m.contentTask.content
    m.contentGrid.setFocus(true)
end function

function onKeyEvent(key as string, press as boolean) as boolean
    if press then
        if key = "back"
            return false ' Let system handle back key
        else if key = "OK"
            playSelectedContent()
            return true
        end if
    end if
    return false
end function
```

### Content Grid Component

```
<!-- ContentGrid.xml -->
<component name="ContentGrid" extends="RowList">
    <interface>
        <field id="content" type="node" />
        <field id="selectedItem" type="node" />
    </interface>

    <script type="text/brightscript">
        <![CDATA[
        function init()
            m.top.itemComponentName = "GridItem"
            m.top.numRows = 3
            m.top.rowFocusAnimationStyle = "fixedFocusWrap"
            m.top.vertFocusAnimationStyle = "fixedFocus"
        end function
        ]]>
    </script>
</component>
```

## Video Player Integration

```
' VideoPlayer.brs
function init()
    m.video = m.top.findNode("videoPlayer")
    m.video.observeField("state", "onVideoStateChange")
    m.video.observeField("position", "onVideoPositionChange")
end function

function playContent(contentNode as object)
    videoContent = createObject("roSGNode", "ContentNode")
    videoContent.url = contentNode.url
    videoContent.title = contentNode.title
    videoContent.streamFormat = "hls" ' or "mp4", "dash"

    ' DRM configuration if needed
    if contentNode.drm <> invalid then
        videoContent.encodingType = contentNode.drm.type
        videoContent.encodingKey = contentNode.drm.key
    end if

    m.video.content = videoContent
    m.video.control = "play"
end function
```

# API Integration

## Content API Service

```
' ContentApi.brs
function getContentFeed() as object
    request = createObject("roUrlTransfer")
    request.setUrl("https://api.dr33mtv.com/content/feed")
    request.addHeader("Content-Type", "application/json")
    request.addHeader("Authorization", "Bearer " + getAuthToken())

    response = request.getToString()
    if response <> "" then
        return parseJson(response)
    end if

    return invalid
end function

function getRecommendations(userId as string) as object
    request = createObject("roUrlTransfer")
    request.setUrl("https://api.dr33mtv.com/ai/recommendations")
    request.addHeader("Content-Type", "application/json")

    postData = {
        "user_id": userId,
        "platform": "roku",
        "preferences": getUserPreferences()
    }

    request.postFromString(formatJson(postData))
    response = request.getToString()

    if response <> "" then
        return parseJson(response)
    end if

    return invalid
end function
```

## AI-Powered Recommendations

```
' AI recommendation integration
function loadAIRecommendations()
    m.recommendationTask = createObject("roSGNode", "RecommendationTask")
    m.recommendationTask.observeField("recommendations", "onRecommendationsLoaded")
    m.recommendationTask.userId = getCurrentUserId()
    m.recommendationTask.control = "RUN"
end function

function onRecommendationsLoaded()
    recommendations = m.recommendationTask.recommendations
    if recommendations <> invalid then
        updateRecommendationRow(recommendations)
    end if
end function
```

## Streaming Features

### HLS Streaming Support

```
function setupHLSStream(streamUrl as string)
    videoContent = createObject("roSGNode", "ContentNode")
    videoContent.url = streamUrl
    videoContent.streamFormat = "hls"

    ' Adaptive bitrate configuration
    videoContent.minBandwidth = 500000    ' 500 Kbps minimum
    videoContent.maxBandwidth = 5000000   ' 5 Mbps maximum

    return videoContent
end function
```

### DRM Integration

```
function setupDRMContent(contentUrl as string, drmConfig as object)
    videoContent = createObject("roSGNode", "ContentNode")
    videoContent.url = contentUrl

    if drmConfig.type = "playready" then
        videoContent.encodingType = "PlayReadyLicenseAcquisitionUrl"
        videoContent.encodingKey = drmConfig.licenseUrl
    else if drmConfig.type = "widevine" then
        videoContent.encodingType = "WidevineLicenseAcquisitionUrl"
        videoContent.encodingKey = drmConfig.licenseUrl
    end if

    return videoContent
end function
```

## Localization

### Multi-Language Support

```
' Localization helper functions
function getLocalizedString(key as string) as string
    locale = getDeviceLocale()
    strings = getStringsForLocale(locale)

    if strings[key] <> invalid then
        return strings[key]
    else
        ' Fallback to English
        englishStrings = getStringsForLocale("en_US")
        return englishStrings[key]
    end if
end function

function getStringsForLocale(locale as string) as object
    ' Load localized strings from locale files
    filePath = "pkg:/locale/" + locale + ".json"
    return readJsonFile(filePath)
end function
```

### Locale Files Structure

```
// locale/en_US.json
{
    "app_title": "Dr33MTV",
    "welcome_message": "Welcome to Dr33MTV",
    "loading": "Loading...",
    "play": "Play",
    "pause": "Pause",
    "search": "Search",
    "recommendations": "Recommended for You",
    "trending": "Trending Now",
    "genres": "Genres",
    "artists": "Artists"
}
```

## Testing and Debugging

### Debug Console Access

```
# Connect to debug console
telnet ROKU_IP 8085

# Common debug commands
print "Debug message"
? variable_name
list
cont
step
```

### Performance Monitoring

```
# Connect to performance monitor
telnet ROKU_IP 8080

# Monitor memory usage, CPU, etc.
```

### Testing Checklist

- [ ] Channel loads correctly
- [ ] Navigation works with remote control
- [ ] Video playback functions properly
- [ ] Audio quality is acceptable
- [ ] UI is responsive
- [ ] Memory usage is within limits
- [ ] Network errors are handled gracefully

## Deployment and Distribution

### Development Deployment

```bash
# Quick deployment script
#!/bin/bash
ROKU_IP="192.168.1.XXX"
ROKU_PASSWORD="your_password"

# Package the channel
zip -r Dr33MTV_Channel.zip . -x "*.git*" "deploy.sh" "README.md"

# Deploy to Roku
curl -s -S -F "mysubmit=Install" -F "archive=@Dr33MTV_Channel.zip" \
     -u "rokudev:$ROKU_PASSWORD" \
     "http://$ROKU_IP/plugin_install"

echo "Channel deployed successfully!"
```

### Production Submission

#### 1. Channel Store Requirements

- **Channel Icons**: HD (290x218) and SD (214x144) PNG files
- **Screenshots**: Multiple screenshots showing channel functionality
- **Channel Description**: Detailed description of channel features
- **Content Rating**: Appropriate content rating
- **Privacy Policy**: Required for channels collecting user data

#### 2. Submission Process

1. **Package Channel**: Create final production package
2. **Developer Portal**: Submit via Roku Developer Portal
3. **Review Process**: Roku reviews channel for compliance
4. **Certification**: Channel must pass technical and content review
5. **Publication**: Channel becomes available in Roku Channel Store

### Channel Store Metadata

```json
{
    "channel_name": "Dr33MTV",
    "description": "Revolutionary global music streaming with AI-powered discovery",
    "long_description": "Dr33MTV brings you the world's music through advanced AI re-
commendations...",
    "category": "Music",
    "content_rating": "All Audiences",
    "languages": ["English", "Spanish", "French", "German", "Japanese"],
    "countries": ["US", "CA", "UK", "AU", "DE", "FR", "ES", "JP"],
    "keywords": ["music", "streaming", "AI", "global", "discovery"]
}
```

## Advanced Features

### Voice Search Integration

```
function handleVoiceSearch()
    if m.top.hasVoiceRemote then
        m.voiceDialog = createObject("roSGNode", "VoiceDialog")
        m.voiceDialog.observeField("text", "onVoiceSearchResult")
        m.voiceDialog.show = true
    end if
end function

function onVoiceSearchResult()
    searchQuery = m.voiceDialog.text
    performSearch(searchQuery)
end function
```

### Deep Linking Support

```
function handleDeepLink(args as object)
    if args.contentId <> invalid then
        ' Launch specific content
        loadAndPlayContent(args.contentId)
    else if args.search <> invalid then
        ' Perform search
        performSearch(args.search)
    end if
end function
```

### Analytics Integration

```
function trackEvent(eventName as string, properties as object)
    analyticsData = {
        "event": eventName,
        "properties": properties,
        "timestamp": createObject("roDateTime").asSeconds(),
        "device_id": getDeviceId(),
        "channel_version": getChannelVersion()
    }

    sendAnalytics(analyticsData)
end function
```

## Troubleshooting

### Common Issues

#### Channel Won’t Load

- Check manifest file syntax
- Verify all required assets are present
- Ensure proper file permissions

#### Video Won’t Play

- Verify stream URL is accessible

- Check video format compatibility
- Test DRM configuration if applicable

### Performance Issues

- Monitor memory usage
- Optimize image sizes
- Reduce concurrent network requests

### Network Connectivity

- Implement proper error handling
- Add retry logic for failed requests
- Provide user feedback for network issues

## Debug Techniques

```
' Debug logging
function debugLog(message as string)
    print "[DEBUG] " + message
    ' Also send to remote logging service if needed
end function

' Memory monitoring
function checkMemoryUsage()
    memInfo = createObject("roDeviceInfo").getGeneralMemoryLevel()
    if memInfo = "critical" then
        ' Handle low memory situation
        freeUnusedResources()
    end if
end function
```

## Performance Optimization

### Memory Management

- Dispose of unused objects promptly
- Limit concurrent video streams
- Optimize image loading and caching
- Monitor memory usage regularly

### Network Optimization

- Implement proper caching strategies
- Use appropriate video bitrates
- Handle network errors gracefully
- Implement offline capabilities where possible

### UI Responsiveness

- Use efficient list components
- Implement lazy loading for large datasets
- Optimize image rendering
- Minimize UI updates during video playback

## Additional Resources

- Roku Developer Documentation (https://developer.roku.com/docs)
- BrightScript Language Reference (https://developer.roku.com/docs/references/brightscript/language)
- SceneGraph Framework (https://developer.roku.com/docs/developer-program/core-concepts/scenegraph)
- Roku Channel Store Guidelines (https://developer.roku.com/docs/developer-program/publishing)
- Performance Best Practices (https://developer.roku.com/docs/developer-program/performance-guide)

---

For technical support or questions, refer to the main project documentation or contact the development team.