Creating Tizen Native Apps with the Native UI & Graphics Framework

Danny Whang
Samsung Electronics
Agenda

- Overview
- UI and Graphics
- More Features
- Tools
Overview
Overview

- The UI & Graphics Framework provides
  - Hierarchy of controls and containers
  - 2D and 3D Graphics with effects & animation
  - Customization with themes
Hello world!
Architectural View

Web Applications

Tizen Web Framework

- W3C / HTML5
- Video
- Touch
- CSS3
- WebGL
- Worker
- Device APIs
  - BT
  - Call
  - NFC
  - Msg
- Web UI F/W
- Web Runtime

Tizen Native Framework

- App / Shell
- Graphics / UI
- Content
- Telephony
- Base / Io
- Uix
- Social
- Messaging
- Text / Locales
- Web / Xml
- Security
- Net
- System
- Media

Core

- App Framework
- Graphics / UI
- Location
- Multimedia
- Messaging
- Web
- Security
- System
- Base
- Connectivity
- Telephony
- PIM

Linux Kernel and device drivers
Detailed View

### Tizen Native UI & Graphics Framework

- Tizen::Ui::Animations
- Tizen::Ui::Effects
- Tizen::Ui::Scenes
- Tizen::Ui::Controls
- Tizen::Ui::Ime
- Tizen::Graphics
- Tizen::Graphics::Opengl

### Core Graphics & UI

- Evas / ECore
- Cairo
- Freeetype2
- Fontconfig
- EGL
- OpenGL ES 1.1/2.0
- X Window System / Window Manager (E17)

---

- Linux Kernel
- Input devices
- Frame buffer
UI & Graphics
UI & Graphics

- **UI**
  - Hello Tizen
  - Controls
  - Animation
  - Visual Element

- **Graphics**
  - 2D Canvas
  - 3D OpenGL ES
  - Canvas Texture
  - Video Texture
Hello Tizen

- Simple Form-based app containing
  - Frame
  - Indicator
  - Form with a Header, Footer, and Button

- To create the app, create a new project using the Form-based application template
Basic Components

• **UI controls**
  - Functional unit of UI
  - Certain controls called containers can contain other controls

• **Frame**
  - Container of Forms

• **Form**
  - Logical unit of the UI Workflow
  - Base container of most controls
Write Your Own Form

• To create a Form
  • Add a Form subclass
  • Implement Form::OnInitializing()
  • Implement Form::OnTerminating()

Class MyForm : public Tizen::Ui::Controls::Form
{
  Public:
    MyForm(void) {};
    virtual ~MyForm(void) {};
  
  Public:
    virtual result OnInitializing(void);
    virtual result OnTerminating(void);
Make Your Form Work

• To make the Form functional, add it to a Frame and set it as the current Form

    // Create a form in the heap, no need to delete explicitly later
    MyForm* pForm = new MyForm();

    // Construct the form
    pForm->Construct(FORM_STYLE_INDICATOR | FORM_STYLE_HEADER | FORM_STYLE_FOOTER);

    // Get the application frame
    Frame* pFrame = UiApp::GetInstance()->GetFrameAt(0);

    // Add the form to the frame
    pFrame->AddControl(pForm);

    // Set your form as the current form
    pFrame->setCurrentForm(pForm);

    // Draw
    pFrame->Invalidate(true);
Handle UI Events

• Inherit the event listener interface

```
Class MyForm : public Tizen::Ui::Controls::Form,
              public Tizen::Ui::IAActionEventListener
{
  Public:
    virtual result OnInitializing(void);
    virtual result OnTerminating(void);
    virtual void OnActionPerformed(const Tizen::Ui::Control& source, int actionId);
}
```

• Register the event handler

```
result MyForm::OnInitializing(void) {
    pButton->SetActionId(ID_BUTTON_BACK);
    pButton->AddActionEventListener(*this);
    AddControl(pButton);
}
```
Handle UI Events

• Implement the event handler

```cpp
void MyForm::OnActionPerformed(const Control& source, int actionId)
{
    switch(actionId)
    {
    case ID_BUTTON_BACK:
        // Handle the button back (ID_BUTTON_BACK) event
        break;
    }
}
```
Animation

- **To add animation between 2 Forms**
  1. Get the animator (*FrameAnimator* or *ControlAnimator*)
  2. Set parameters and call *SetCurrentForm()* for the transition

```cpp
// Get FrameAnimator for a form transition
FrameAnimator* pAnimator = pCurrentFrame->GetFrameAnimator();

// Set up animation parameters
pAnimator->SetFormTransitionAnimation(
    FRAME_ANIMATOR_FORM_TRANSITION_ANIMATION_TRANSLATE_RIGHT,  // Animation type
    500,  // Duration
    ANIMATION_INTERPOLATOR_LINEAR  // Interpolation
);

// Change to a new form; transition animation starts automatically
r = pAnimator->SetCurrentForm(pNextForm);
```
Animation Explained

• Basic animation contains
  • Start value
  • Key value (created automatically using interpolators)
  • End value

• Animation can be applied to the following properties
  • Position
  • Size
  • Alpha
  • Rotation
Animation Classes

- Classes
  - IntegerAnimation
  - FloatAnimation
  - PointAnimation
  - DimensionAnimation
  - RectangleAnimation
  - RotateAnimation

- Event listener
  - IControlAnimatorEventListener
    - OnControlAnimationStarted()
    - OnControlAnimationStopped()
    - OnControlAnimationFinished()
Animation Example

- To animate a Button from one position to another
  - Initialize the UI controls
  - Use point animation to create an animation

```cpp
case ID_BUTTON:
{
    result r;
    ControlAnimator* pButtonAnimator = __pButton->GetControlAnimator();

    Point startPos = __pButton->GetPosition();
    Point endPos(startPos.x, startPos.y + 200);

    PointAnimation pointAnimation(startPos, endPos, 2000, ANIMATION_INTERPOLATOR_LINEAR);
    pointAnimation.SetAutoReverseEnabled(true);

    r = pButtonAnimator->StartUserAnimation(ANIMATION_TARGET_POSITION, pointAnimation);
} break;
```
Visual Element

- A conceptual 2D rectangular model for animation and composition
  - Transforms 2D plane in 3D space (2.5D)
  - Property-based architecture with support for implicit and explicit animations
  - GPU accelerated
Property

- Most APIs can be accessed via the SetProperty() and GetProperty() methods

```csharp
pVE = new VisualElement();
pVE->Construct();

pVE->SetProperty (L"bounds", FloatRectangle(0.0f, 0.0f, 100.0f, 100.0f));
pVE->SetBounds (FloatRectangle(0.0f, 0.0f, 100.0f, 100.0f));
```

- Bounds / Opacity / Show state / Anchor
- Transform Rotate[X|Y|Z], Scale[X|Y|Z], Translation[X|Y|Z]
- Z-position / Z-order group
- Custom property can be added
- Animation is as easy as changing the property
Graphics

- Canvas is a memory buffer where all drawing happens

```cpp
result MyForm::OnDraw(void) {
    ...
    Canvas* pCanvas = GetCanvasN();
    pCanvas->Clear();
    pCanvas->SetForegroundColor(...);
    pCanvas->DrawEllipse(...);
    ...
    delete pCanvas;
    return r;
}
```
# 2D Drawing Primitives

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Line style</th>
<th>Fill style</th>
<th>Composite mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Width</td>
<td>Join style</td>
<td>Cap style</td>
</tr>
<tr>
<td>Pixel</td>
<td>--</td>
<td>--</td>
<td>-</td>
</tr>
<tr>
<td>Line</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polyline</td>
<td>0</td>
<td></td>
<td>Round Miter Bevel</td>
</tr>
<tr>
<td>Triangle</td>
<td>0</td>
<td>Round Butt Square</td>
<td>0</td>
</tr>
<tr>
<td>Rectangle</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arc</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polygon</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ellipse</td>
<td>0</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td>--</td>
<td>--</td>
<td>-</td>
</tr>
<tr>
<td>Bitmap</td>
<td>--</td>
<td>--</td>
<td>-</td>
</tr>
</tbody>
</table>
OpenGL® ES

- EGL
- OpenGL ES 1.1, 2.0

```cpp
bool GlesSample::InitEGL()
{
    EGLint numConfigs = 1;
    EGLint eglConfigList[] = { /*...*/};
    EGLint eglContextList[] = { /*...*/};

    eglBindAPI(EGL_OPENGL_ES_API);
    eglDisplay = eglGetDisplay((EGLNativeDisplayType)EGL_DEFAULT_DISPLAY);
    eglInitialize(eglDisplay, null, null);
    eglChooseConfig(eglDisplay, eglConfigList, &eglConfig, 1, &numConfigs);
    eglSurface = eglCreateWindowSurface(eglDisplay, eglConfig, (EGLNativeWindowType)pForm, null);
    ...
}
```
Canvas Texture

- Utility for mapping Canvas to Texture

```cpp
// Initialize Canvas Texture
glGenTextures(1, &__texture);
__pCanvasTexture = new CanvasTexture;
__pCanvasTexture->Construct(__texture, 1280, 720);

Canvas* pCanvas = __pCanvasTexture->GetCanvasN();

Font font;
font.Construct(FONT_STYLE.PLAIN, 200);
pCanvas->SetFont(font);

pCanvas->Clear();
pCanvas->DrawText(Point(offset, 500), L"Canvas");
pCanvas->DrawText(Point((-offset, 700), L"Texture");

// Draw a frame with the texture
glBindTexture(GL_TEXTURE_2D, __texture);

glDrawElements(GL_TRIANGLES, numIndices, GL_UNSIGNED_SHORT, INDICES);
```
Video Texture

• Utility for mapping Video to Texture

```cpp
// Initialize Video Texture
glGenTextures(1, &__texture);
__pVideoTexture = new VideoTexture;
__pVideoTexture->Construct(__texture, 1280, 720);

// This function gets the IVideoTextureUpdateListener
__pVideoTexture->SetVideoTextureUpdateListener(*this);

__pPlayer = new Tizen::Media::Player();
__pPlayer->Construct(*this, __pVideoTexture);
__pPlayer->OpenFile(L"data/Helicopter.mp4");
__pPlayer->Play();

// Draw a frame with the texture
__pVideoTexture->BindTexture();
glDrawElements(GL_TRIANGLES, numIndices, GL_UNSIGNED_SHORT, INDICES);
```
More Features
More Features

- **Tizen::Ui**
  - Accessibility
  - Downloadable IME
  - Effect Manager
  - Scalable UI
  - Scene Manager

- **Tizen::Shell**
  - Notification Manager
  - Notification Tray
  - Dynamic Box
Accessibility

• Large Font

```cpp
// Retrieve font size from user setting
Tizen::System::SettingInfo::GetValue(
    L"http://tizen.org/setting/font.size", fontSizeString);
fontSize = Tizen::Ui::UiConfiguration::GetFontSize(fontSizeString);
```

• Screen reader

```cpp
// Make an accessibility element for custom drawing
AccessibilityElement* pAccessibilityElement = new AccessibilityElement();
pAccessibilityElement->Construct(GetBounds(), L"Tizen Image");
pAccessibilityElement->SetLabel(L"Tizen Image");
pAccessibilityElement->SetTrait(L"Image");
pAccessibilityElement->SetHint(L"This image rotates automatically.");
GetAccessibilityContainer()->AddElement(*pAccessibilityElement);
```
Downloadable IME

- You can create custom IME in these short steps
Effect Manager

- Effect model with LUA script-based interaction for 3D transition effects

```cpp
Tizen::Ui::Effects::EffectManager* __pEffectManager;

// Effect instance
Effect* __pEffect;

// Panel for OpenGL surface for effect drawing
Panel* __pEffectsPanel;

// IEffectEventListener
virtual void OnEffectStarted(Tizen::Ui::Effects::Effect& effect);
virtual void OnEffectFinished(Tizen::Ui::Effects::Effect& effect,
                              Tizen::Ui::Effects::EffectResult effectResult,
                              const Tizen::Base::Collection::IList& lastShownBitmapIds);

// IEffectResourceProvider
virtual result SetBitmap(Tizen::Ui::Effects::Effect& effect, long bitmapId);
```
Scalable UI

- Logical coordinate system – 480, 720, etc.
- Layout manager – relative, linear (H & V), grid, card

- Bitmap (density) and XML layout (screen size) fallback
Scene Manager

- Simpler Scene-based navigation instead of Forms
Scene Manager

- Register scenes

```cpp
SceneManager* pSceneManager = SceneManager::GetInstance();
pSceneManager->RegisterScene(L"Scene1", L"Form1", L"Form1Panel");
pSceneManager->RegisterScene(L"Scene2", L"Form2", L"Form2Panel");
```

- Scene transition

```cpp
pSceneManager->GoForward(ForwardSceneTransition(L"Scene1");
pSceneManager->GoForward(ForwardSceneTransition(L"Scene2", SCENE_TRANSITION_ANIMATION_TYPE_NONE, SCENE_HISTORY_OPTION_NO_HISTORY));
pSceneManager->GoBackward(BackwardSceneTransition(L"Scene1", SCENE_TRANSITION_ANIMATION_TYPE_RIGHT));
```
Notification

• Send notifications to the user easily

```cpp
Tizen::Shell::NotificationManager notificationMgr;
notificationMgr.Construct();

notificationMgr.Notify(L"A new message has arrived");
notificationMgr.NotifyTextMessage(L"Even simpler message");
notificationMgr.NotifyOngoingActivity(L"Something new is in progress");
```

• NotificationManager requires privilege, add this in the manifest.xml editor
Custom Control for Notification Tray

- Add UI Controls to the QuickPanelFrame

```cpp
// Create QuickPanelFrame
_pQuick = new Tizen::Shell::QuickPanelFrame();
_pQuick->Construct(400.0f);

// Add controls
_pQuick->AddControl(_pSlider);
_pQuick->AddControl(_pButton);

// Change the show state in order to change the visibility
_pQuick->SetShowState(true);
_pQuick->Show();

// Hide
_pQuick->SetShowState(false);
_pQuick->Show();
```
Dynamic Box

• Dynamic Box is a small app that can be embedded in other apps such as the Home screen

• SDK contains host Viewer sample app (using Tizen::Shell::AppWidgetView) and AppWidget app template for you to start with
Dynamic Box

- Complexity is hidden behind `AppWidgetProvider`

```cpp
bool MyAppWidgetProvider::OnAppWidgetProviderInitializing(float width, float height, const Tizen::Base::String& userInfo)
{
    // Initialize AppWidgetFrame and AppWidgetProvider specific data
    AppWidgetFrame* pFrame = new MyAppWidgetFrame();
    pFrame->Construct(Dimension(width, height));
    this->SetAppWidgetFrame(pFrame);
    pFrame->Show();
    return true;
}

bool MyAppWidgetProvider::OnAppWidgetProviderUpdating(const Tizen::Base::String& argument)
{
    // Update Dynamic Box
    pAppWidgetFrame->Invalidate();
    return true;
}
```
Even More Features
Even More Features

- **Tizen::Uix**
  - Sensor
  - Speech
  - Vision

- **Tizen::App**
  - AppControl
  - AppResource

  - Motion
  - Light
  - Proximity
  - Accelerometer
  - Gyro
  - Face detect
  - Face recognize
  - Image object
  - QR code

- **Tizen::Web**
  - Web control

- **Tizen::Media**
  - Image encoder and decoder
  - Audio & video encoder and decoder
  - Audio & Video player
  - Camera

- **Even More Features**
  - Tizen::Web
  - Tizen::Media
Tools
Tools

• UI Builder
• Effect Builder
• UI Customizer
UI Builder

• XML authoring
• Code generator
• Orientation
Effect Builder

- Effect in XML and LUA script
- Realistic 3D with physics
  - Page flipping
  - 3D rotation
  - Scrolling
UI Customizer

- Application can embed a theme
- OEM or carrier can offer custom default theme