Tizen 3D UI
DALi 3D Engine
building exciting User Interfaces

Kimmo Hoikka
Samsung

TIZEN™
DEVELOPER CONFERENCE
2014
SAN FRANCISCO

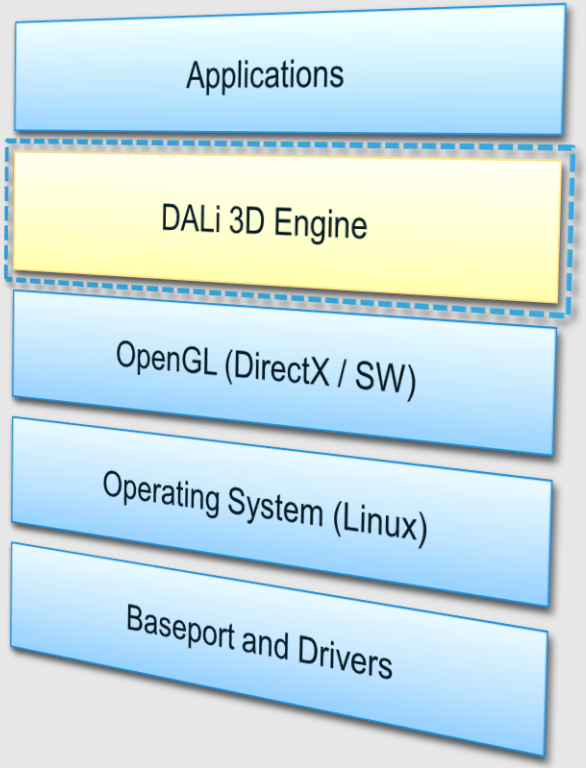
Introduction

Introduction

- **Kimmo Hoikka**
  - Head of 3D UI & Graphics Middleware team in Samsung Electronics R&D UK
  - 17 years in commercial SW development, past 15 years in Mobile UI & Graphics, Middleware domains
  - Before commercial career 10 years of Graphics Demo programming Amiga 500, Commodore C64, etc

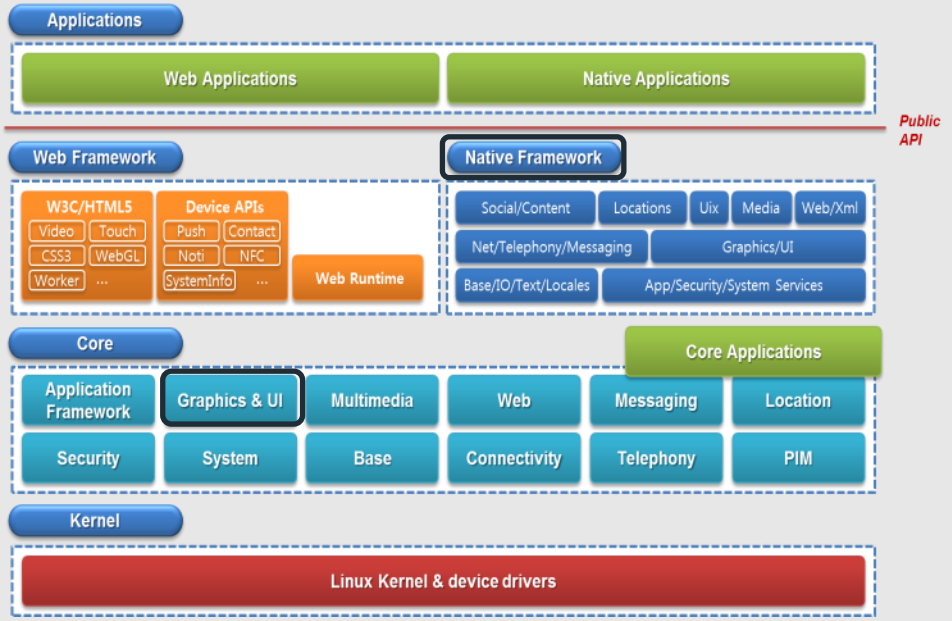
Introduction

- **Tizen 3D UI**
  - DALi 3D Engine & UI Toolkit
- **DALi is a 3D Engine**
  - UI is represented as a **3D Scene Graph**
  - **Animations** and **Transitions** are done using **3D Math** (Vector, Quaternion & Matrix)
  - **Rendering** and **Visual Effects** are done using Open GL ES Shaders, Vertices and Textures
  - OpenGL ES 2 and 3 support
- **2D world is the Z plane 0 in the 3D world**
  - When using default camera



System Architecture

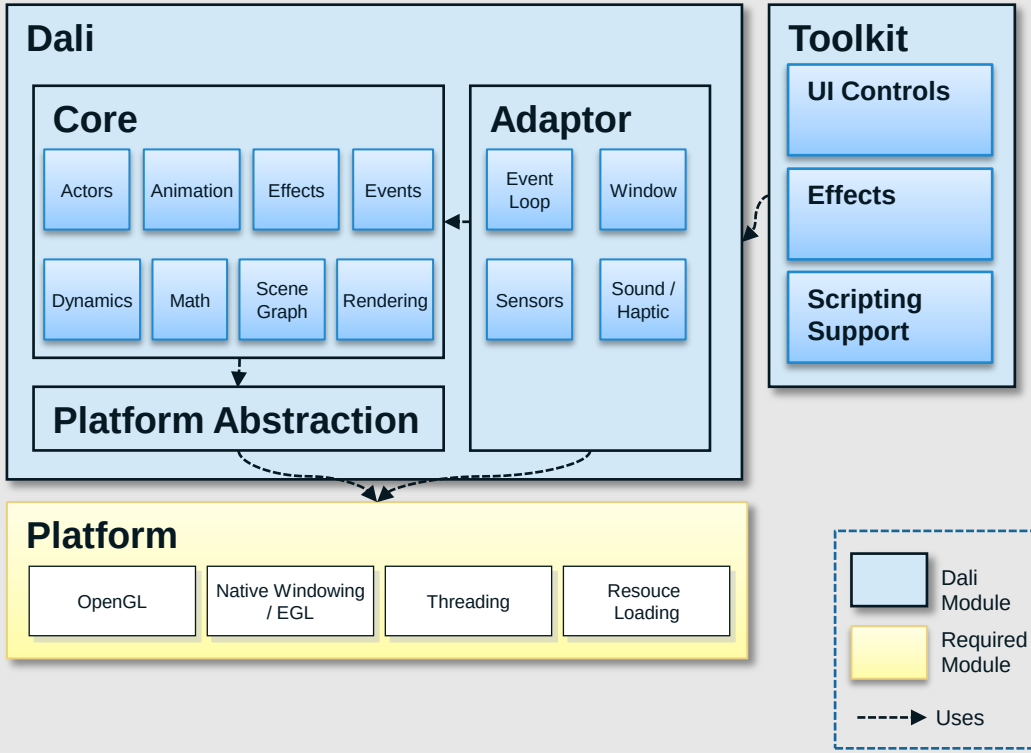
- **DALi is part of the Tizen Native Framework**
  - Graphics & UI Core module
  - Mobile and TV profiles
- **Implemented in C++**
- **DALi (Dynamic Animation Library)**
  - 2D and 3D Application UIs with Realistic Effects & Animations
  - Home Screen, Lock Screen, Gallery, Music Player …



Architecture

Architecture

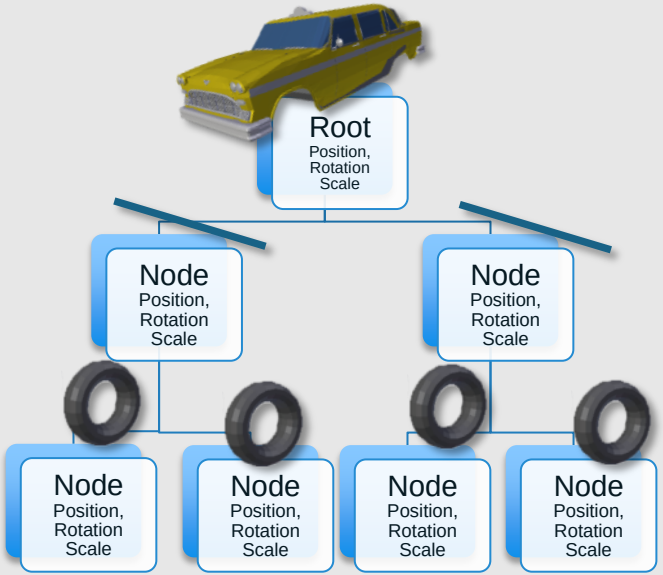
- **Core Library**
  - Event handling, Scene Graph, Rendering, Resource management
- **Adaptor**
  - Threading model
  - Integration with the main loop
- **Platform abstraction**
  - Resource loading and decoding with multiple threads
- **Toolkit**
  - Reusable UI controls,
  - Effects and Scripting support



3D Scene Graph

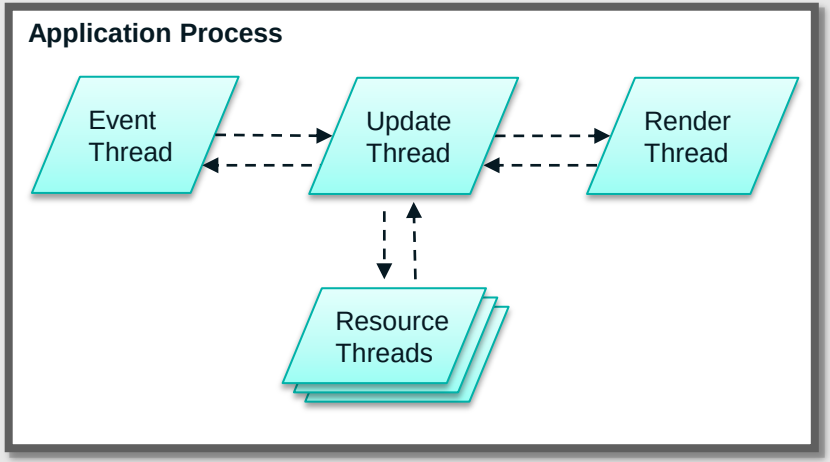
- **Scene graph based UI is a tree of Nodes**
  - Each Node can have 0-N Children
  - Each Node inherits its parent Transformation

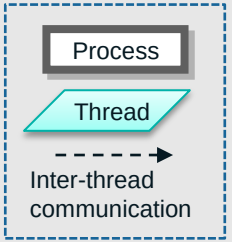
  $ **`Position, Rotation, Scale`**
    - Allows easy layout and animation management
  - Each Node's Transformation is relative to a reference point in the parent's space
    - Anchor point in the Nodes own coordinate space
    - Parent origin in the Parents coordinate space
  - Child does not have to be inside its parent area



Multithreaded Engine

- **DALi uses multithreaded architecture**
  - Best performance and scalability
- **Event Thread**
  - The main thread in which application code and event handling runs
- **Update Thread**
  - Updates the nodes on scene
  - Runs animations, constraints and physics
- **Render Thread**
  - Open GL drawing, texture and geometry uploading etc
- **Resource Threads**
  - Loads font, image and model resources and decodes into bitmaps etc





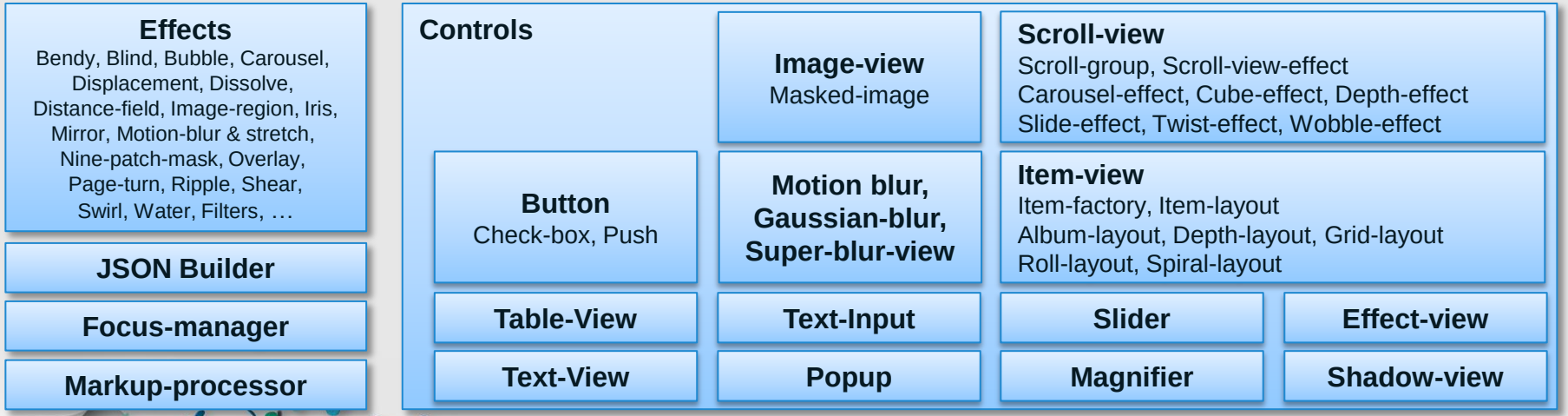
3D Core library

- Animation framework
- Event & gesture handling
- Rendering of the 3D scene
- Physics plug-in API
- Model loading plug-in API
- Core is platform and window system agnostic

Actors Camera, Image, Layer, Light, Mesh, Text	**Animation** Alpha-functions, Constraint Key Frames	**Common** Stage, Light	**Dynamics** Body, Collision, Joint Shape, World
Event & Gesture Touch, Key, Mouse-wheel / Tap, LongPress, Pan, Pinch	**Geometry** Mesh, Spline, Animated-mesh/vertex,	**Images** Bitmap, Distance-field, Frame buffer, Native	**Math** Matrix, Quaternion, Radian, Vector, Rect
Modeling Bone, Entity, Material, Model-animation	**Render-tasks** On/Off Screen, Viewport	**Shader-effects** Uniform animation	**Text** Font, Text-Style

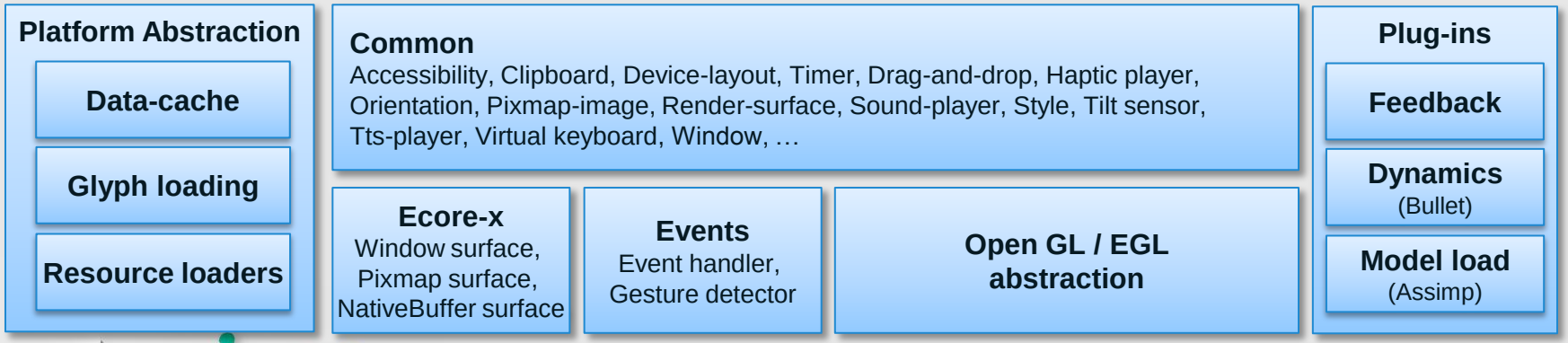
3D Toolkit library

- **Full Application UI development facilities**
  - UI Controls, such as Buttons, Text view …
  - Effects, such as Page turn, Motion blur
  - Focus management, Accessibility, Styling support etc
- **JSON Builder**
  - Defining UI in an external JSON file produced by GUI builder or by developer



Adaptor libraries

- **Application framework and Window system integration**
  - Provides integration into the native windowing system: EFL, X11, Wayland…
- **Multithreading control and synchronization**
- **Platform Abstraction isolates the core module from platform specific parts**
  - For example Resource loading and decoding (Images, Glyphs, …)
- **Plug-in implementations for external optional modules**



APIs: C++

- **Applications can be developed in C++**

```
// C++
Dali::ImageActor imageActor = Dali::ImageActor::New( Dali::Image::New( "/photos/background.jpg" ) );
imageActor.SetParentOrigin( Dali::ParentOrigin::CENTER );
imageActor.SetAnchorPoint( Dali::AnchorPoint::CENTER );
Dali::Stage::GetCurrent().Add( imageActor );
...
bool onPressed( Dali::Actor, const TouchEvent& event )
{
  Dali::Animation anim = Dali::Animation::New( 1.5f );
  anim.MoveTo( actor, Vector3( 200,-100,0), AlphaFunctions::Bounce );
  anim.play();
  return true; // consume the touch event
}
...
imageActor.TouchedSignal().Connect( &onPressed );
```

APIs: JavaScript

- **Applications can be developed in JavaScript (*)**

```
// JavaScript
var imageActor = new dali.ImageActor( new dali.Image( "/photos/background.jpg" ) );
imageActor.parentOrigin = dali.CENTER;
imageActor.anchorPoint = dali.CENTER;
dali.stage.add( myImageActor );
...
function onPressed( actor, touchEvent )
{
  var animOptions = { alpha: "bounce", delay: 0, duration: 15 };
  var anim = new dali.Animation();
  anim.animateTo( actor, "position", [ 200,-100,0], animOptions );
  anim.play();
  return true; // consume the touch event
}
...
imageActor.connect( "touched", onPressed );
```

(*) under development

APIs: JSON

- Application UI layout and interaction can also be described in JSON

// JSON

```
"animations":
 {
   "move-image":
   {
     "duration": 1.5,
     "properties":
     [
       {
         "actor":"image",
         "property":"position",
         "value":[200,-100,0],
         "alpha-function": "BOUNCE",
       }
     ]
   }
 }
```

```
"stage":
  [
    {
      "name":"image",
      "type":"ImageActor",
      "image":
      {
        "filename":"/photos/background.jpg"
      },
      "signals" :
      [
        { "name" : "touched", "action": "play",
          "animation": "move-image" }
      ],
    }
  ]
}
```

Features

Features: Actors & UI Controls

- **Stage is the root of the world**
  - Actors are processed when they are on-stage
- **Image, Text and Mesh Actors are the Building Blocks (*)**
  - Built-in properties include Position, Size, Rotation, Scale, ParentOrigin, AnchorPoint and Color
- **UI Controls provide additional Layouting and Scrolling**
  - Buttons, Sliders, Popup etc as basic UI controls
  - ScrollView, ItemView for Scrolling of contents or views
  - Alignment, TableView, Navigation frame etc for traditional layouting & UI hierarchy management

(*) Particle Actor under development

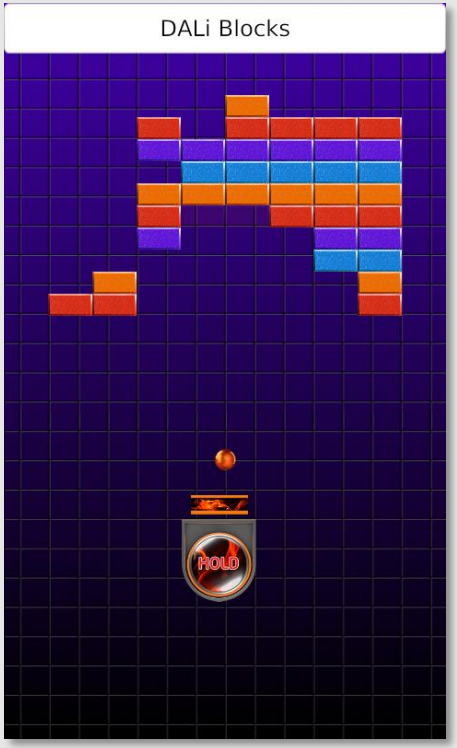
Features: Animation

- **Property animation**
  - Predefined actor properties (Position, Size, Scale, Rotation, Color, Visibility)
  - Custom properties (Added by Application or UI Control)
- **Vertex & Mesh animation**
  - Deform mesh (for example animated graphs)
- **Shader Uniform animation**
  - Control the shader effect
- **Model animation**
  - Bone & Joint animation
- **Key frame animation**
- **Flexible system**
  - Single animation can contain properties from many objects
  - Animations will blend if the target property is same

Features: Constraints and Property Notifications

- **Constraint**
  - Allows making a property a function of other properties

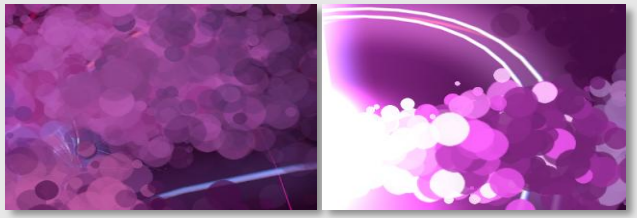
  $ `Property = Func(property1,property2,…)`
  - In breakout example, Collision property is a function of Position of ball, Position of paddle, Size of the ball and Size of the paddle
  - Constraint function can calculate when the ball hits a paddle and set collision property to true
- **Property notification**
  - Application can get notification when property crosses a threshold or reaches a value
  - In the breakout example, when collision is true; ball changes direction and sound is played



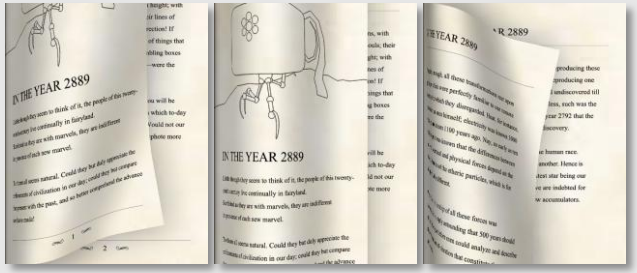
Features: Shader Effects

- **Shader effects can modify the appearance of objects during rendering**
  - Each Actor has its own default Shader based on its geometry type (Image, Text and Mesh)
  - Geometry (vertex) or Pixels (fragment) or both can be modified by overriding the default shader
- **Lots of built-in Shaders in Toolkit**
  - Bendy, Blind, Bubble, Carousel, Displacement, Dissolve, Distance-field, Image-region, Iris, Mirror, Motion-blur & stretch, Nine-patch-mask, Overlay, Page-turn, Ripple, Shear, Swirl, Water, Filters, …

Dissolve Effect



Bubble Effect



Page Turn Effect

Features: Effects

- **Image effects**
  - Cube transition effects: Cross, Fold, Wave
- **Effect containers**
  - Containers that apply an effect for all its children
    - Bloom effect
    - Gaussian Blur
    - Super blur
    - Shadow View
    - Effect View
- **Bubble effect**
- **Motion blur effect**

Cube Transition Effect

Shadow View

Motion Blur Effect

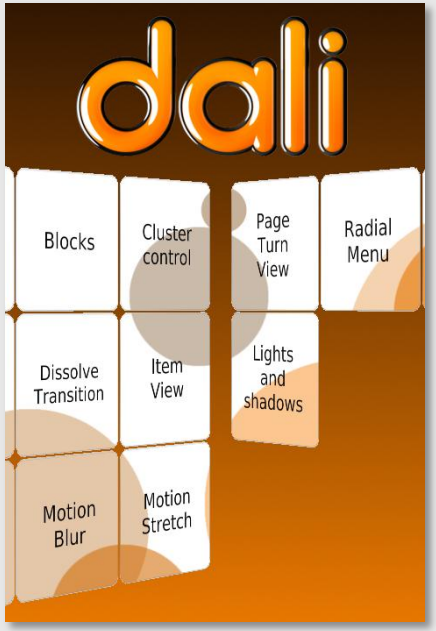
Features: ItemView

- **ItemView**
  - Scrolling container based on data source provided by application
  - Layout specifies each items layout using constraints and items layout position
    - Constraint for Position, Size, Color, Rotation, Scale, …
  - Built in layouts: Grid, Spiral, Depth, Album, Navigation, Roll
  - Application can provide custom layout
  - Layout can be dynamically changed, all items are animated automatically to new layout.

ItemView layouts: Grid, Depth, Spiral

Features: ScrollView

- **ScrollView**
  - Scrolling container with Scroll effect support
  - Horizontal & Vertical scrolling
  - Flick, Snap, Axis lock, Custom Rulers
  - Does not layout its children, just moves them
  - Built in Scroll-effects
    - Carousel, Cube, Depth, Twist, Page Cube, Slide, Wobble, …



Inner cube scroll effect

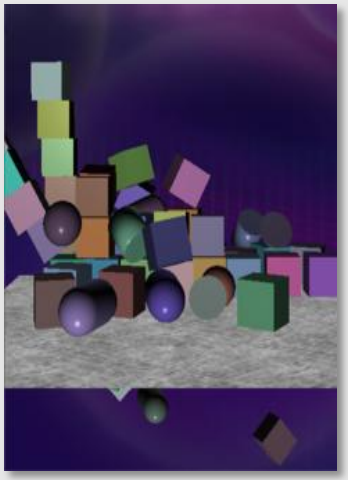
Features: 3D Models & Bone animation

- **Model loading support**
  - Industry standard formats, e.g. Collada, Maya, 3DS, etc
  - Own Binary format (faster start-up)
- **Model importer plug-in**
  - Uses Open Asset Import Library (assimp) to load industry standard models.
- **Bone and key-frame animations also supported from models**

Model Import and
Key-frame Animation

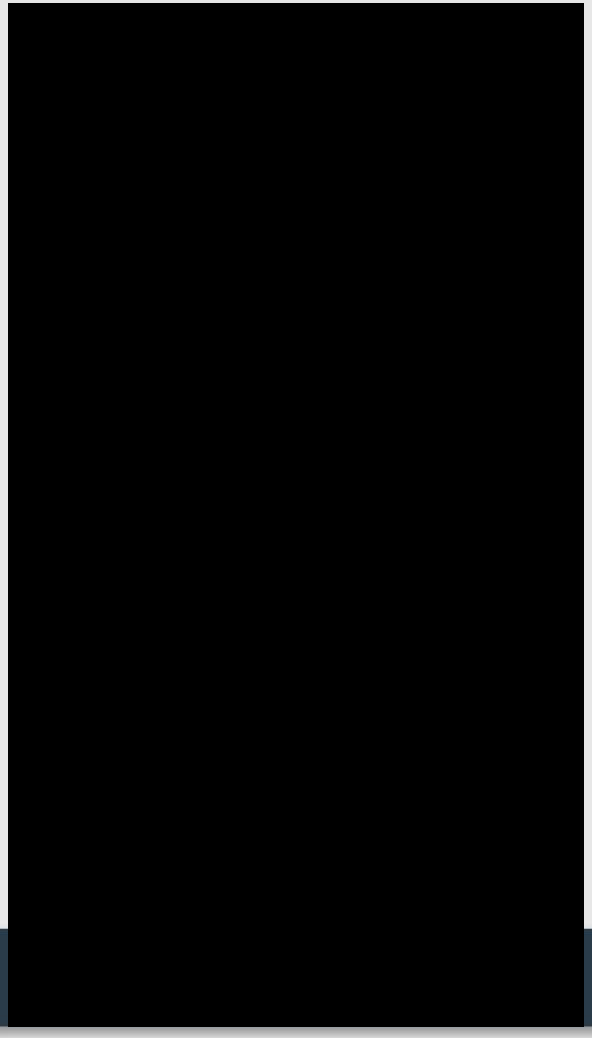
Features: Physics integration

- **Supports rigid and soft body physics effects**
  - Actor has dynamics API to set properties for the physics simulation
  - Actor::EnableDynamics()
    The actor will behave as a rigid/soft body in the simulation
  - Stage::InitializeDynamics()
    Initialize the dynamics world and enable simulation
- **Physics is a plug-in API**
  - Allows integrating any third party physics engine
  - Bullet plug-in provided with adaptor



Rigid body
collision example

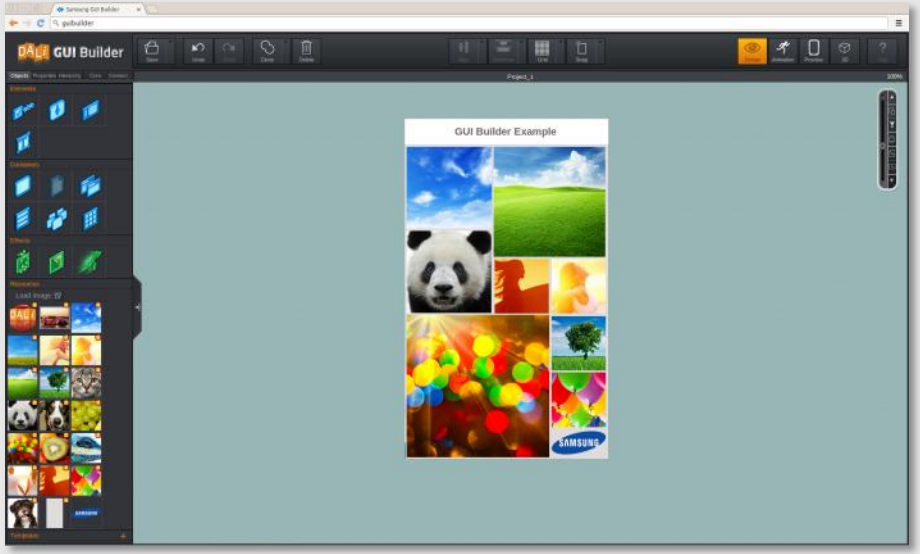
Features: Video



Features: GUI Builder

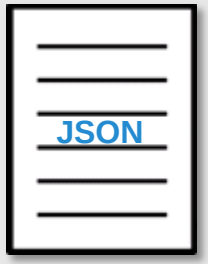
- **DALi provides scripting support**
  - Creating a scene using a variety of actors
  - Creating animations for actor properties: position, rotation, size etc.
  - Changing the style of an actor
  - Scriptable functionality is described in a JSON file
- **GUI Builder is an interactive, visual tool to create a UI**
  - Browser based, so naturally cross platform
    - Uses a combination of HTML, CSS & JavaScript
  - Outputs a JSON file that DALi-launcher can run or C++ application can load

GUI Builder: Static Layout development

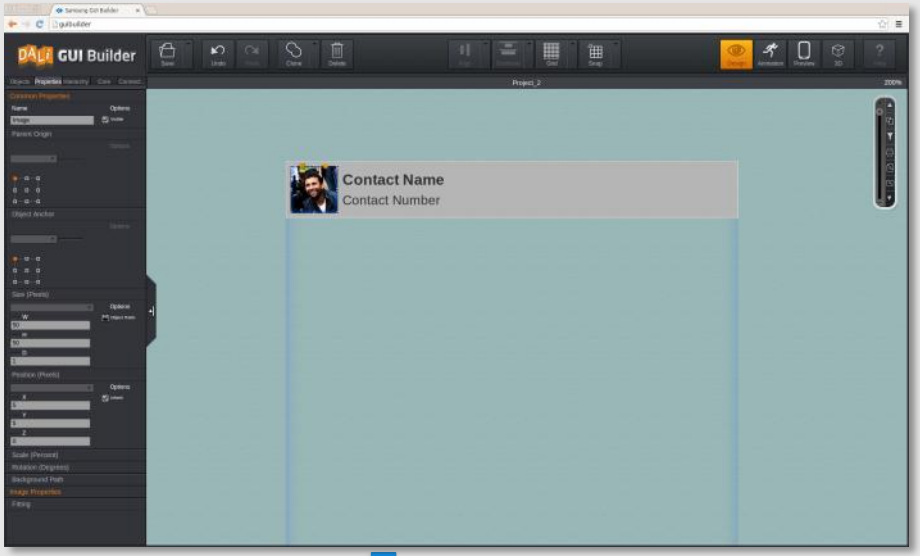


Run with
Dali-launcher

Output a Script

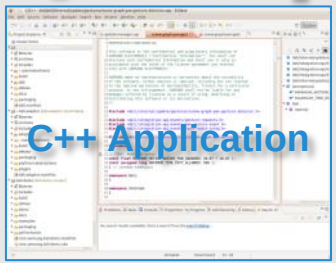


GUI Builder: C++ Application with JSON layouts



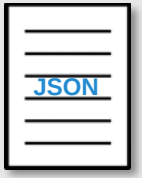
Retrieve contacts

Run

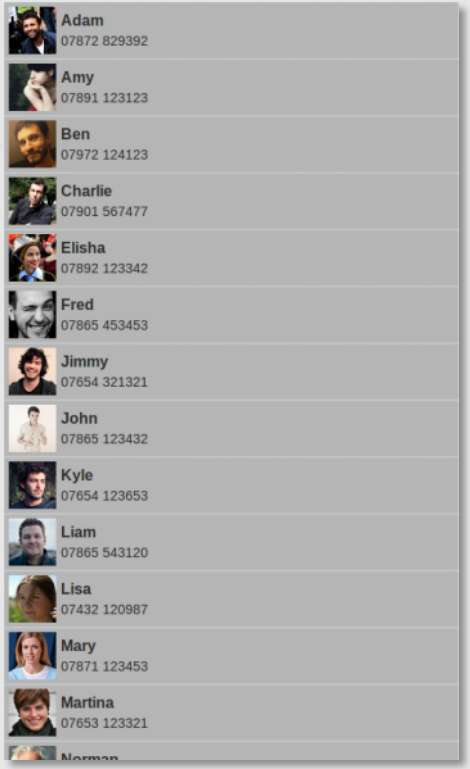


Load template(s) to use, apply data

Export Template(s)



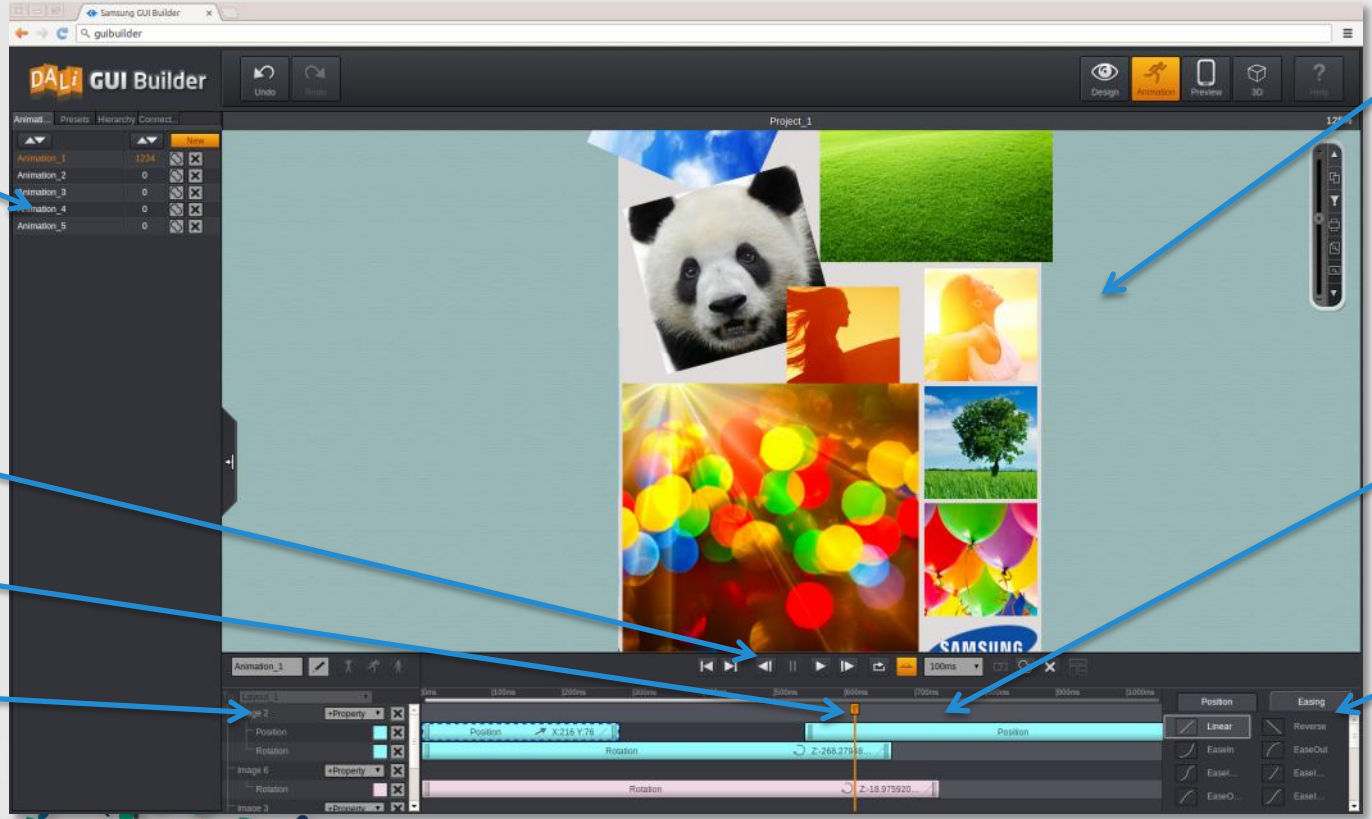
Loaded at Runtime



GUI Builder: Animation View

- **Directly manipulate the scene to create animations**
  - Drag & drop to create move animations
  - Resize to create size animations
  - Scale & rotate to create scale & rotation animations
- **Editable & interactive timeline**
  - Movable playback head, easily add & combine animation segments
- **Connect Animations with Actions (e.g. button-press)**
- **A variety of easing functions for the animations**
  - Linear, Sine, Ease In, Bounce etc.

GUI Builder: Animation View



GUI Builder: Video

What next?

- **Get the code:**
  - git clone https://review.tizen.org/gerrit/platform/core/uifw/dali-core
  - git clone https://review.tizen.org/gerrit/platform/core/uifw/dali-toolkit
  - git clone https://review.tizen.org/gerrit/platform/core/uifw/dali-adaptor
  - git clone https://review.tizen.org/gerrit/platform/core/uifw/dali-demo
- **Play with it**
  - Build Cool and Exciting applications !!!
- **Contribute**
  - Ideas, Features, Bug fixes !!!

Thank You!!

Contact: kimmo *<dot>* **hoikka** *<at>* **samsung** *<dot>* **com**


TIZEN™
DEVELOPER
CONFERENCE
2014
SAN FRANCISCO