

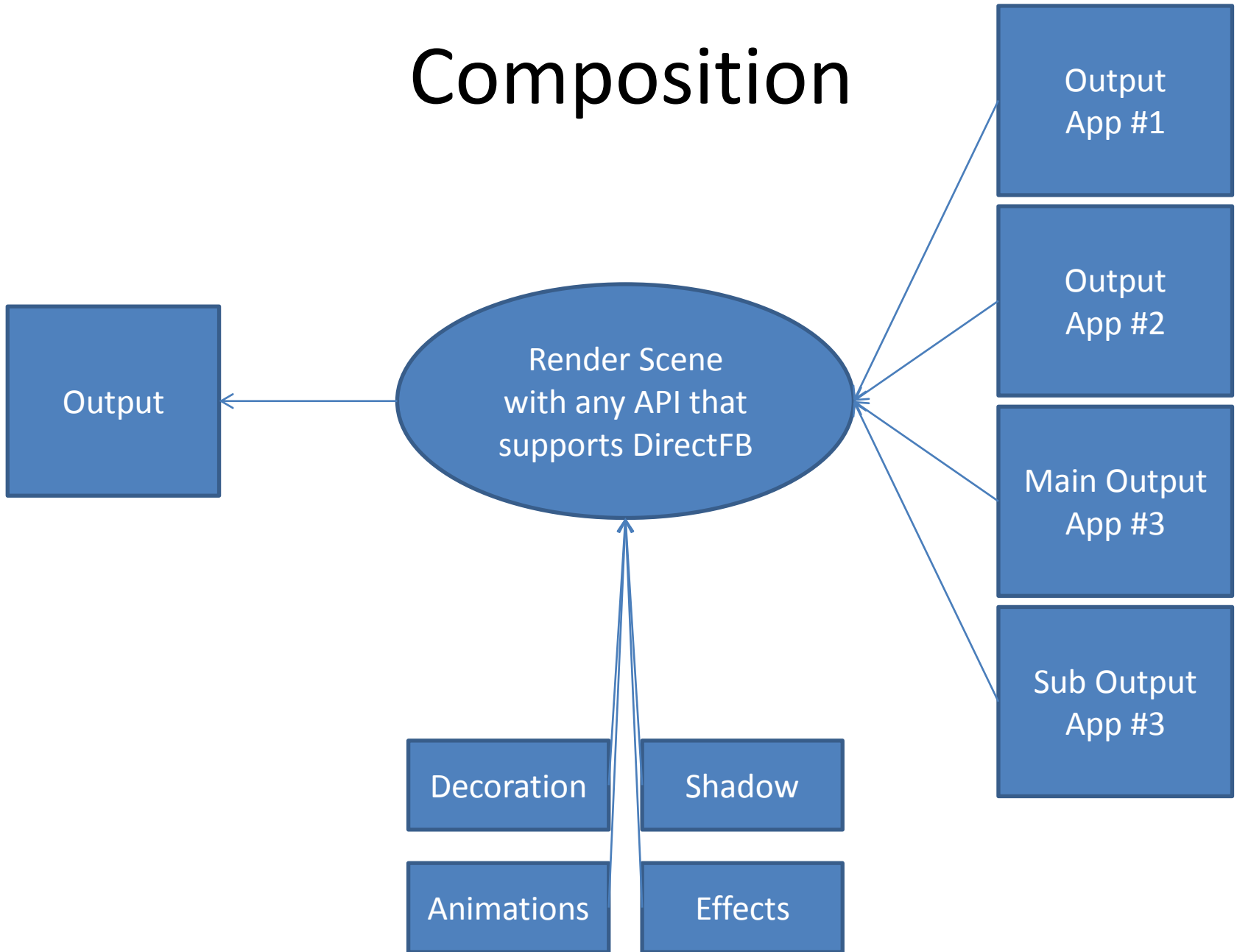
# Compositing Application Manager

How to write a DirectFB Compositor

# Composition

- Compositor has an output surface and any number of input surfaces from other applications.
- Renders the whole scene including its own resources, e.g. decorations around the application views, shadows, animations...
- Updates in an application trigger events in the compositor to schedule next composition.

# Composition



# Compositor Main Loop

- Fetch input and update events from DirectFB
  - Schedule updates for each application view which has received update events from the application
  - Handle input events for decorations and other local objects, dispatch events to applications as appropriate, e.g. being in ‘focus’
- Render to output if updates are scheduled.

# Applications

- Application is a high-level view on a process.
- They can have lots of meta information and attached resources like output surfaces with layout information.
- Any process can register as an application.

# Surfaces

- An application creates a surface and registers that as an output in the application registry.
- Compositors are notified about registration.
- Application surfaces (identified by ID) are obtained by a compositor via a call to
  - `IDirectFB::GetSurface( ID )`
- and attached to its event buffer via
  - `IDirectFBSurface::AttachEventBuffer( buffer )`.

# Update Events

- DFBSurfaceEvent is triggered each time the application owning the surface calls
  - IDirectFBSurface::Flip( region ) or
  - IDirectFBSurface::FlipStereo( left, right ).
- Each event contains the region(s) of the call.
- The compositor schedules an update for the affected region within its own scene, e.g. a portion of a widget showing the app surface.

# Rendering

- Applications and compositors can use any rendering API supporting DirectFB
  - IDirectFBSurface's native 2D API
  - OpenGL, OpenVG
  - Cairo
  - ...