Ecosystem TSG Update
Jon Leech, TSG Chair

Pipeline Newsletter

- Will get back on a regular publication schedule
- Will give more information about specs and extensions – once we're sure it's accurate

Future Ecosystem Projects

- **Update man pages for 3.0**
  - Can leverage some of the OpenGL ES 2.0 work
- **Bring conformance tests into the shader era**
  - Start with ES 2.0 tests
  - Add coverage for new API features
  - New tests for GLSL 1.30 features

API Evolution – Some Terms

- **Core**
  - All features in a version of the Specification
- **Deprecate**
  - Mark a feature for reduction in support
  - Will be identified in the Specification
  - Features are not removed in the version in which they're deprecated
- **Profile**
  - A well-defined subset of the core
  - Only the ARB may define profiles
  - Driven by market requirements
  - We found a requirement for only one profile in OpenGL 3.0
    - Future **possibility**: “entertainment” and “content creation”
    - Future **possibility**: “desktop” and “mobile”
    - No stated intent to do either, these are examples
  - Deprecation is profile-specific

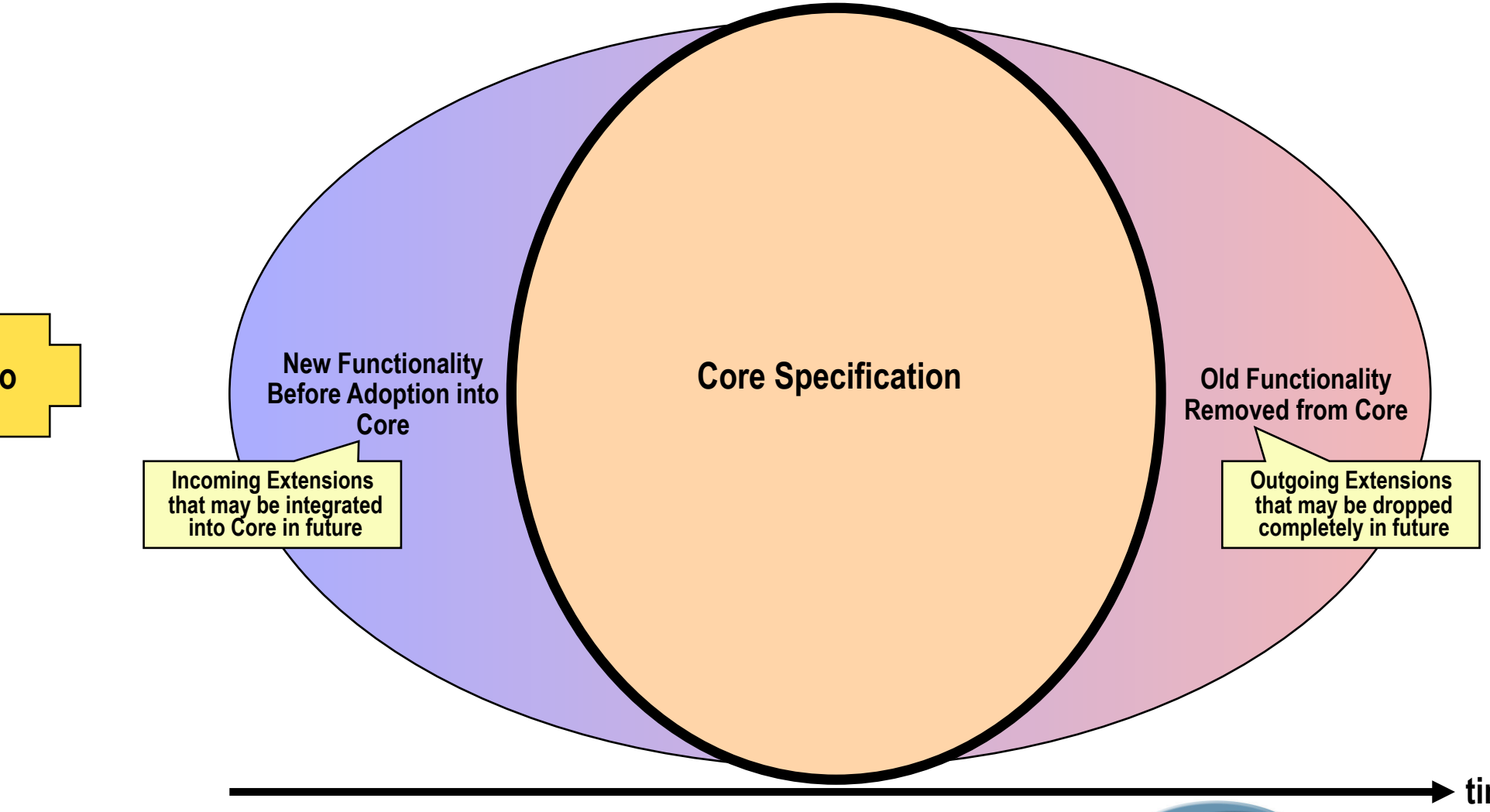
Feature Removal Phases

- **Stage 0: Core**
  - In core, fully supported
  - **Will** be present in the next API version
- **Stage 1: Core (Deprecated)**
  - In core, marked as deprecated
  - **May** be fully or partly removed in a later version
  - New features need not define interactions with deprecated ones
- **Stage 2: Extension**
  - **Removed** from core, becomes an ARB extension (no suffixes)
  - Extension spec identifies the removed functionality
  - Vendors may support the extension if their markets require it
- **Stage 3: Removed**
  - No longer an ARB extension (for this and later API versions)
  - Could be an EXT or vendor extension, if vendor markets still require it (still no suffixes required)

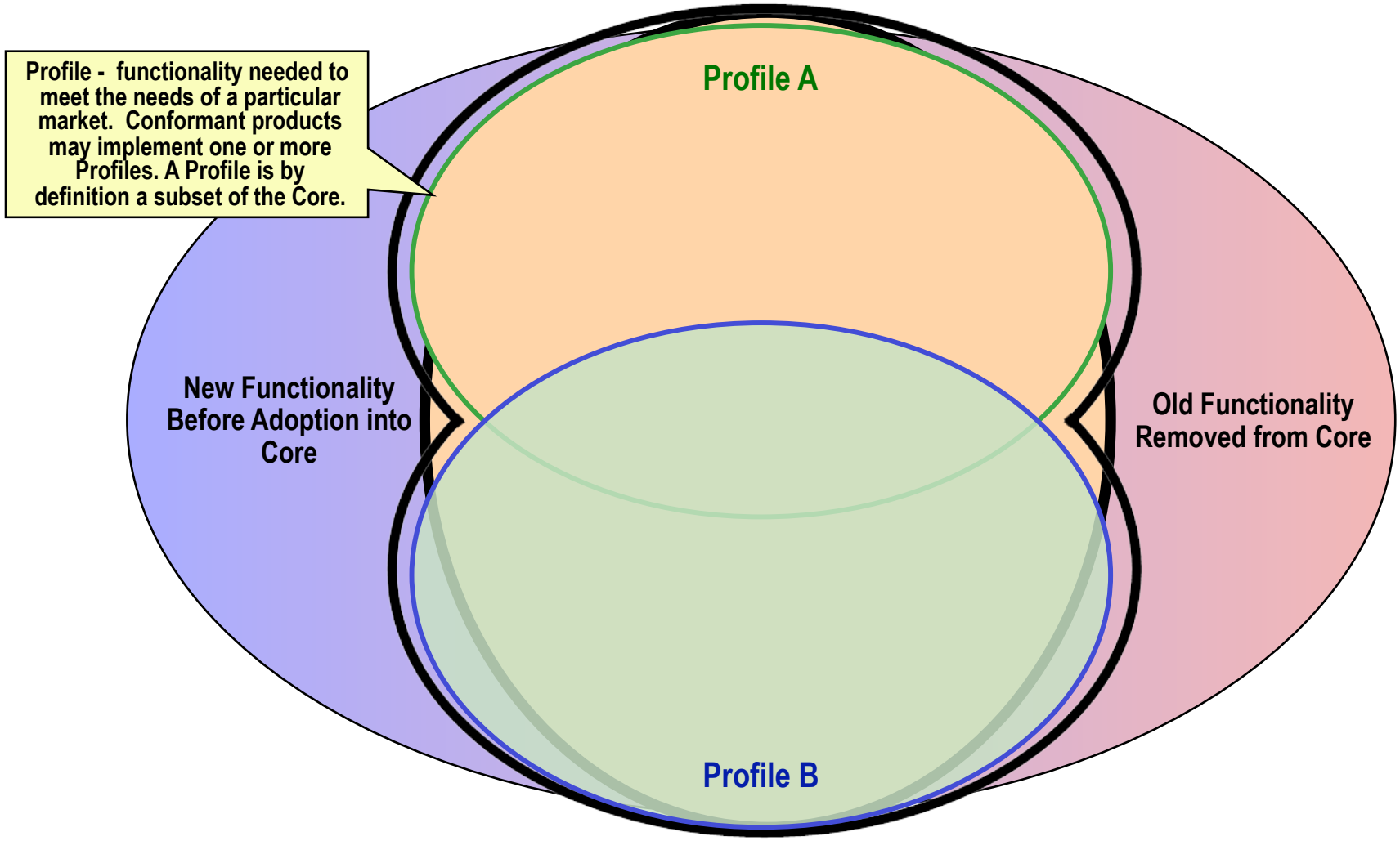
Feature Removal

- Features will be deprecated for at least one release (stage 1) before being removed
- Extension Path: `Vendor/EXT->ARB->Core`
  - With possible API / functionality changes as we learn from experience
- Deprecation Path: `Core->ARB->EXT/Vendor`
  - No API or functionality changes

Evolution Model - Deprecation

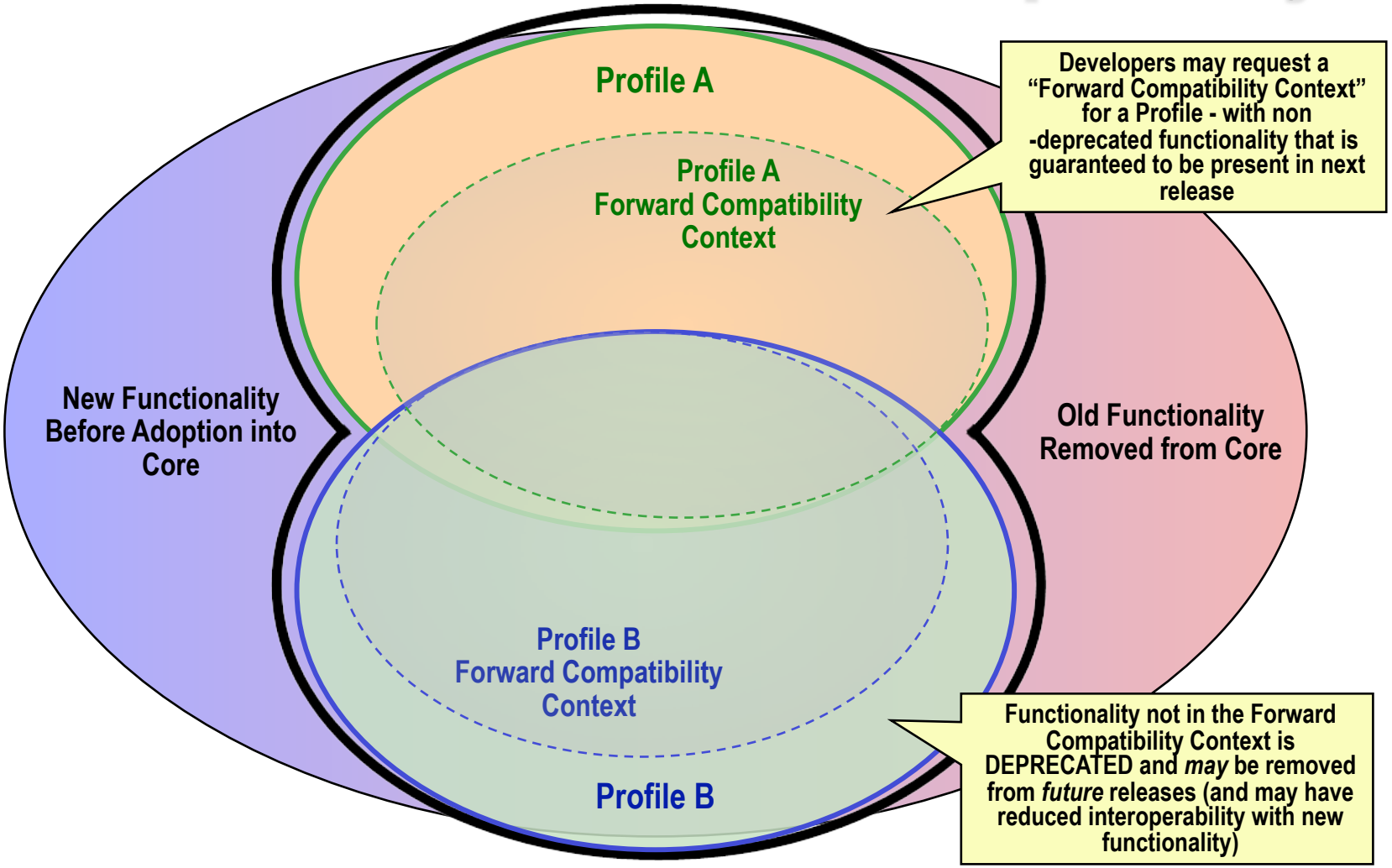


Evolution Model - Profiles



Core Specification
The sum of all Profile functionality

Evolution Model – Forward Compatibility



Core Specification
The sum of all Profile functionality

Context & Feature Subsets

- **"Full" contexts contain all features defined in that version and profile**
- **"Forward-compatible" contexts contain only the non-deprecated features in that version and profile of the API**
  - May include extensions if those extensions do not restore deprecated functionality
  - Intended as a coding aid
  - Not a "preview". Only shows what may be taken away, not what may be coming

Context Creation

- In the past creating a context gave you whatever version the driver decided
- The API was always backwards compatible, so there was no harm in getting a later version
- Starting with OpenGL 3.1, backwards compatibility may (almost certainly will) no longer exist due to deprecation
- Apps need a way to specify which functionality they require when creating a context

WGL_ARB_create_context

- **wglCreateContextAttribs(hDC, hShareContext, const int *attribList)**
- **attribList contains { token, value } pairs such as**
  - { MAJOR_VERSION, major}
  - { MINOR_VERSION, minor}
  - { CONTEXT_FLAGS, flags }
  - Legacy information (e.g. layer planes)
- **Version number**
  - major.minor <= 2.1
    - Returned version >= requested, <= 2.1 (backwards compatibility)
  - major.minor == 3.0
    - Returned version must be exactly 3.0
  - major.minor > 3.0
    - TBD but probably must return exactly major.minor (unless we someday define a 3.(N+1) which is a strict superset of 3.N)
  - Default value is 1.0 (typically an OpenGL 2.1 context is returned)

WGL_ARB_create_context

- **Flags is a bitmask of modifiers possibly including**
  - CONTEXT_FORWARD_COMPATIBLE_BIT
    - Return a forward-compatible context
  - CONTEXT_DEBUG_BIT
    - Return a debug context
- **If multiple profiles are defined in the future, flags or attributes to identify those profiles will be added to context creation**
- **Legacy context creation calls cannot return 3.0+ contexts**
  - Existing 2.1 apps would break once features start being deprecated
  - Start asking for what you require
- **3.0 contexts may be made current without drawables**
  - Supports “headless rendering”
- **GLX_ARB_create_context coming very shortly**
- **Other platforms (MacOS X) will have equivalents**

App Compatibility

- **Contexts created with a specific version, profile, and forward-compatible bit**
  - Source compatible across all implementations
  - Binary compatible on the same OS / window-system binding platform (Windows/WGL, Linux/GLX, etc.)
- **Context of a later version and the same profile**
  - Compatible with apps that don't use features not in the later version
  - API Specification will define which earlier versions (if any) are fully source-compatible.
  - Code that runs on a deprecated context of version 3.N is always compatible with a full context of version 3.(N+1)
- **Similar to how GLSL handles multiple versions**