This is an overview of the features of the next release of Lightning, provisionally dubbed version 1.1. Version 1.2, possibly 1.3, and 2.0 (using full OpenGL) are additionally planned.

The plan is for a new major release every 4 to 8 weeks, but **this may change.**

Feature requests should be supplied fairly quickly for them to end up in this version.

**Warning: This list is NOT an exhaustive list of changes and more may be made at any time between now and the next release!**

**Delta Time**  
All areas of the engine will obey delta time. Any area of the engine that uses frames will use milliseconds (msec) in the future. **This feature may be backported to version 1.0 in a 1.0.1 release.**

**SDL**  
SDL will be updated to version 2.24.0, or whatever the latest version is when 1.1 is released.   
All other SDL dependencies will be updated, as well as functions added to bindings.

**New Animation Engine**  
Lightning 1.1 will support a new animation engine that will allow developers to animate any visible properties of a Renderable with any cycle. Keyframes will be supported.

AnimatedTexture will be rewritten to inherit from Texture.

**Cameras**  
A new camera type will be added that positions the player at the bottom of the screen.  
Camera shake may be implemented.

**Lighting**  
Coloured lighting will be implemented.

**Input**  
The **Key::Either\*\*\*\*Required** methods will be made static in order to make them less awkward to use.

**Packaging**Lightning will verify the engine version in a package file’s header and will not load a package intended for an older version of the engine.

**Localisation**It will become possible to set a folder for localisation files.  
It will become possible to set the **GlobalSettings::Localisation** file at runtime.

**System Information**The **SystemInformation::ScreenResolutionX** and **SystemInformation::ScreenResolutionY** parameters will support all monitors instead of just the first one.

**Text**  
SDL2\_ttf will be replaced with custom freetype2 bindings that support caching text glyphs to reduce memory usage, add OTF support and increase performance. The Text Manager API will be overhauled to be a state machine – modifying the properties of TextManager before drawing text (which will only have one parameter, **cWindow**) will change how the text appears. Font loading will work similarly to how it currently does.

The old font APIs will be supported in 1.1, but marked obsolete and will be removed in 1.2.

**TextureManager**  
TextureManager will be overhauled and advanced with various new features:

**Texture Caching**Textures will be cached in order to increase performance and reduce memory usage.  
Any number of textures can refer to the same image.

**New Rendering API**Textures will be plugged into TextureManager, instead of being rendered manually in the render loop. Their properties

**UI**The UI code will be cleaned up and redundant functionality removed.  
Multi-line textboxes will be implemented.