Документация по продукту Lambda-Mu Community Edition Version 3

Туториалы

Платформа Лямбда-Мю 3 + Android-приложение

- Многопользовательский режим (чат-комната)
- Мультиязычность
- Мастер-экземляр платформы ("балансировщик" нагрузки)

Платформа Лямбда-Мю 3 + Godot Engine

- 3D/VR многопользовательский режим
- Голосовой чат
- Ментор

Платформа Лямбда-Мю 3 + Unreal Engine

- Подключение платформы

Технические характеристики

Установка

Первые шаги

How-To

Иснтрументы

Проекты сообщества

Разное

.. _doc_list_of_features:

List of features

This page aims to list all features currently supported by Godot.

.. note::

```
This page lists features supported by the current stable version of
Godot (3.2). `More features
<https://docs.godotengine.org/en/latest/about/list_of_features.html>`__
are available in the latest development version (4.0).
```

Features

Platforms

Can run both the editor and exported projects:

- Linux (64-bit and 32-bit, x86 and ARM).

Binaries are statically linked and can run on any distribution if compiled on an old enough base distribution. Official binaries are compiled on Ubuntu 14.04.

Runs exported projects:

- Android 4.4 and later. - iOS 10.0 and later. - HTML5 via WebAssembly (Firefox, Chrome, Edge, Opera). - :ref:`Consoles <doc_consoles>`.

Godot aims to be as platform-independent as possible and can be ported to new platforms with relative ease.

Editor

- Support for external script editors such as

```
Visual Studio Code or Vim.
```

- GDScript :ref:`debugger <doc_debugger_panel>`.

1. No support for debugging in threads yet.

- Performance monitoring tools. - Live script reloading. - Live scene editing.

1. Changes will reflect in the editor and will be kept after closing the running project.

- Remote inspector.

1. Changes won't reflect in the editor and won't be kept after closing the running project.

- Live camera replication.

1. Move the in-editor camera and see the result in the running project.

- Use the editor in dozens of languages contributed by the community.

Plugins:

- Editor plugins can be downloaded from the

```
:ref:`asset library <doc_what_is_assetlib>` to extend editor functionality.
```

- Create your own plugins using GDScript to add new features or speed up your workflow. - Download projects from the asset library in the project manager and import them directly.

2D graphics

Two renderers available:

- OpenGL ES 3.0 renderer (uses OpenGL 3.3 on desktop platforms).

1. High-end visuals. Recommended on desktop platforms.

- OpenGL ES 2.0 renderer (uses OpenGL 2.1 on desktop platforms).

1. Recommended on mobile and Web platforms.

Features:

- Sprite, polygon and line rendering.

1. High-level tools to draw lines and polygons such as Polygon2D and Line2D.

- AnimatedSprite as a helper for creating animated sprites. - Parallax layers.

1. Pseudo-3D support by automatically duplicating a layer several times.

- 2D lighting with normal maps.

1. Hard or soft shadows.

- Font rendering using bitmaps (BitmapFont) or rasterization using FreeType (DynamicFont).

1. Bitmap fonts can be exported using tools like BMFont.
2. DynamicFont supports monochrome fonts as well as colored fonts. Supported formats are TTF and OTF.

1. DynamicFont supports optional font outlines with adjustable width and color.
2. Support for font oversampling to keep fonts sharp at higher resolutions.

- GPU-based particles with support for custom particle shaders. - CPU-based particles.

2D tools

- Path2D node to represent a path in 2D space.

2. PathFollow2D node to make nodes follow a Path2D.

- Line2D node to draw textured 2D lines.

Physics bodies:

Collision detection:

- Collision polygons (can be drawn manually or generated from a sprite in the editor).

3D graphics

1. High-end visuals. Recommended on desktop platforms.

- OpenGL ES 2.0 renderer (uses OpenGL 2.1 on desktop platforms).

1. Recommended on mobile and Web platforms.
2. LDR rendering for greater compatibility.
3. Not all features are available. Features available only when using the OpenGL ES 3.0 renderer are marked *GLES3* below.

Camera:

- Perspective, orthographic and frustum-offset cameras.

Physically-based rendering:

- Follows the Disney PBR model. - Uses a roughness-metallic workflow with support for ORM textures. - Normal mapping. - *GLES3:* Parallax/relief mapping with automatic level of detail based on distance. - *GLES3:* Sub-surface scattering and transmittance. - *GLES3:* Proximity fade (soft particles). - Distance fade which can use alpha blending or dithering to avoid going through

```
the transparent pipeline.
```

- Dithering can be determined on a per-pixel or per-object basis.

Real-time lighting:

- Directional lights (sun/moon). Up to 4 per scene. - Omnidirectional lights. - Spot lights with adjustable cone angle and attenuation.

Shadow mapping:

- *DirectionalLight:* Orthogonal (fastest), PSSM 2-split and 4-split.

```
Supports blending between splits.
```

- *OmniLight:* Dual paraboloid (fast) or cubemap (slower but more accurate).

```
Supports colored projector textures in the form of panoramas.
```

- *SpotLight:* Single texture.

Global illumination with indirect lighting:

- Baked lightmaps (fast, but can't be updated at run-time).

1. Lightmaps are baked on the CPU.

- *GLES3:* GI probes (slower, semi-real-time). Supports reflections.

Reflections:

- *GLES3:* Voxel-based reflections (when using GI probes). - Fast baked reflections or slow real-time reflections using ReflectionProbe.

```
Parallax correction can optionally be enabled.
```

- *GLES3:* Screen-space reflections. - Reflection techniques can be mixed together for greater accuracy.

Sky:

- Panorama sky (using an HDRI). - Procedural sky.

Fog:

- Depth fog with an adjustable attenuation curve. - Height fog (floor or ceiling) with adjustable attenuation. - Support for automatic depth fog color depending on the camera direction

```
(to match the sun color).
```

- Optional transmittance to make lights more visible in the fog.

Particles:

- *GLES3:* GPU-based particles with support for custom particle shaders. - CPU-based particles.

Post-processing:

- Tonemapping (Linear, Reinhard, Filmic, ACES). - *GLES3:* Automatic exposure adjustments based on viewport brightness. - *GLES3:* Near and far depth of field. - *GLES3:* Screen-space ambient occlusion. - Glow/bloom with optional bicubic upscaling and several blend modes available:

```
Screen, Soft Light, Add, Replace.
```

- Color correction using an one-dimensional ramp. - Brightness, contrast and saturation adjustments.

Texture filtering:

- Nearest, bilinear, trilinear or anisotropic filtering.

Texture compression:

- *GLES3:* BPTC for high-quality compression (not supported on macOS). - *GLES3:* ETC2 (not supported on macOS). - ETC1 (recommended when using the GLES2 renderer). - *GLES3:* S3TC (not supported on mobile/Web platforms).

Anti-aliasing:

- Multi-sample antialiasing (MSAA).

Most of these effects can be adjusted for better performance or to further improve quality. This can be helpful when using Godot for offline rendering.

Physics: … custom, cylinder, plane, quad, … **Collision detection:** … 3D tools … - CSG … Built-in … Many 2D … and 3D … transforms.

1. PathFollow3D node to make nodes follow a Path3D.
2. 3D geometry helper class.

Collision detection:

- Support for re-mappable :ref:`audio buses <doc_audio_buses>` and effects
- Built-in shapes: line, box, sphere, capsule, cylinder (only with Bullet physics). - Generate triangle collision shapes for any mesh from the editor. - Generate one or several convex collision shapes for any mesh from the editor.

```
with dozens of effects included.
```

- Listener3D node to listen from a position different than the camera in 3D. - Audio input to record microphones. - MIDI input.

1. glTF 2.0 *(recommended)*.
2. `ESCN <https://github.com/godotengine/godot-blender-exporter>`_ (direct export from Blender).

- FBX (experimental). - Collada (.dae). - Wavefront OBJ (static scenes only, can be loaded directly as a mesh).

- Syntax inspired by Python. - Syntax highlighting is provided on GitHub. - :ref:`Use threads <doc_using_multiple_threads>` to perform asynchronous actions

1. No support for MIDI output yet. Input ^^^^^ - Input mapping system using hardcoded input events or remappable input actions. - Axis values can be mapped to two different actions with a configurable deadzone. - Use the same code to support both keyboards and gamepads. - Keyboard input. - Keys can be mapped in «physical» mode to be independent of the keyboard layout. - Mouse input. - The mouse cursor can be visible, hidden, captured or confined within the window. - When captured, raw input will be used on Windows and Linux to sidestep the OS' mouse acceleration settings. - Gamepad input (up to 8 simultaneous controllers). - Pen/tablet input with pressure support. Navigation ^^^^^^^^^^ - A* algorithm in 2D and 3D. - Navigation meshes. - Support for dynamic obstacle avoidance planned in Godot 4.0. - Generate navigation meshes from the editor. Networking ^^^^^^^^^^ - Low-level TCP networking using StreamPeer and TCP_Server. - Low-level UDP networking using PacketPeer and UDPServer. - Low-level HTTP requests using HTTPClient. - High-level HTTP requests using HTTPRequest. - Supports HTTPS out of the box using bundled certificates. - High-level multiplayer API using UDP and ENet. - Automatic replication using remote procedure calls (RPCs). - Supports unreliable, reliable and ordered transfers. - WebSocket client and server, available on all platforms. - WebRTC client and server, available on all platforms. - Support for UPnP to sidestep the requirement to forward ports when hosting a server behind a NAT. Internationalization ^^^^^^^^^^^^^^^^^^^^ - Full support for Unicode including emoji. - Store localization strings using :ref:`CSV <doc_internationalizing_games>` or :ref:`gettext <doc_localization_using_gettext>`. - Use localized strings in your project automatically in GUI elements or by using the ``tr()`` function. - Support for right-to-left typesetting and text shaping planned in Godot 4.0. Windowing and OS integration ^^^^^^^^^^^^^^^^^^^^^^^^^^^^ - Move, resize, minimize, and maximize the window spawned by the project. - Change the window title and icon. - Request attention (will cause the title bar to blink on most platforms). - Fullscreen mode. - Doesn't use exclusive fullscreen, so the screen resolution can't be changed this way. Use a Viewport with a different resolution instead. - Borderless window (fullscreen or non-fullscreen). - Ability to keep the window always on top. - Transparent window with per-pixel transparency. - Global menu integration on macOS. - Execute commands in a blocking or non-blocking manner. - Open file paths and URLs using default or custom protocol handlers (if registered on the system). - Parse custom command line arguments. Mobile ^^^^^^ - In-app purchases on Android and iOS. - Support for advertisements using third-party modules. XR support (AR and VR) ^^^^^^^^^^^^^^^^^^^^^^ - Support for ARKit on iOS out of the box. - Support for the OpenXR and OpenVR APIs. - Popular VR headsets like the Oculus Quest and HTC Vive are supported thanks to plugins. GUI system ^^^^^^^^^^ Godot's GUI is built using the same Control nodes used to make games in Godot. The editor UI can easily be extended in many ways using add-ons. **Nodes:** - Buttons. - Checkboxes, check buttons, radio buttons. - Text entry using LineEdit (single line) and TextEdit (multiple lines). - Dropdown menus using PopupMenu and OptionButton. - Scrollbars. - Labels. - RichTextLabel for :ref:`text formatted using BBCode <doc_bbcode_in_richtextlabel>`. - Trees (can also be used to represent tables). - Containers

```
or make use of multiple processor cores.
```

APIs used:

- *Windows:* WASAPI. - *macOS:* CoreAudio. - *Linux:* PulseAudio or ALSA.

:ref:`C#: <toc-learn-scripting-gdscript>`

Import

- Packaged in a separate binary to keep file sizes and dependencies down. - Uses Mono 6.x.

1. Full support for the C# 7.0 syntax and features.

- Supports all platforms. - Using an external editor is recommended to benefit from IDE functionality.

:ref:`VisualScript: <toc-learn-scripting-visual_script>`

- :ref:`Graph-based visual scripting language <doc_what_is_visual_script>`. - Works best when used for specific purposes (such as level-specific logic)

```
rather than as a language to create entire projects.
```

GDNative (C, C++, Rust, D, ...):

- When you need it, link to native libraries for higher performance and third-party integrations.

1. For scripting game logic, GDScript or C# are recommended if their

performance is suitable.

- Official bindings for C and C++.

1. Use any build system and language features you wish.

- Maintained D, Kotlin, Python, Nim, and Rust bindings provided by the community.

Audio

(horizontal, vertical, grid, center, margin, draggable splitter, ...). - Controls can be rotated and scaled. **Sizing:** - Anchors to keep GUI elements in a specific corner, edge or centered. - Containers to place GUI elements automatically following certain rules. - :ref:`Stack <class_BoxContainer>` layouts. - :ref:`Grid <class_GridContainer>` layouts. - :ref:`Margin <class_MarginContainer>` and :ref:`centered <class_CenterContainer>` layouts. - :ref:`Draggable splitter <class_SplitContainer>` layouts. - Scale to multiple resolutions using the ``2d`` or ``viewport`` stretch modes. - Support any aspect ratio using anchors and the ``expand`` stretch aspect. **Theming:** - Built-in theme editor. - Generate a theme based on the current editor theme settings. - Procedural vector-based theming using :ref:`class_StyleBoxFlat`. - Supports rounded/beveled corners, drop shadows and per-border widths. - Texture-based theming using :ref:`class_StyleBoxTexture`. Godot's small distribution size can make it a suitable alternative to frameworks like Electron or Qt. Animation ^^^^^^^^^^ - Direct kinematics and inverse kinematics. - Support for animating any property with customizable interpolation. - Support for calling methods in animation tracks. - Support for playing sounds in animation tracks. - Support for BÃ©zier curves in animation. Formats ^^^^^^^^ - Scenes and resources can be saved in :ref:`text-based <doc_tscn_file_format>` or binary formats. - Text-based formats are human-readable and more friendly to version control. - Binary formats are faster to save/load for large scenes/resources. - Read and write text or binary files using :ref:`class_File`. - Can optionally be compressed or encrypted. - Read and write :ref:`class_JSON` files. - Read and write INI-style configuration files using :ref:`class_ConfigFile`. - Can (de)serialize any Godot datatype, including Vector, Color, ... - Read XML files using :ref:`class_XMLParser`. - Pack game data into a PCK file (custom format optimized for fast seeking), into a ZIP archive, or directly into the executable for single-file distribution. - :ref:`Export additional PCK files<doc_exporting_pcks>` that can be read by the engine to support mods and DLCs. Miscellaneous ^^^^^^^^^^^^^^ - :ref:`Low-level access to servers <doc_using_servers>` which allows bypassing the scene tree's overhead when needed. - Command line interface for automation. - Export and deploy projects using continuous integration platforms. - `Completion scripts <https://github.com/godotengine/godot/tree/master/misc/dist/shell>`

are available for Bash, zsh and fish.

- Support for :ref:`C++ modules <doc_custom_modules_in_c++>` statically linked

```
into the engine binary.
```

- Engine and editor written in C++03.

1. Can be :ref:`compiled <doc_introduction_to_the_buildsystem>` using GCC,

Clang and MSVC. MinGW is also supported.

1. Friendly towards packagers. In most cases, system libraries can be used

instead of the ones provided by Godot. The build system doesn't download anything.

```
   Builds can be fully reproducible.
 - Godot 4.0 will be written in C++17.
```

- Licensed under the permissive MIT license.

1. Open developement process with :ref:`contributions welcome <doc_ways_to_contribute>`.

.. seealso::

```
The `roadmap <https://github.com/godotengine/godot-roadmap>`__ repository
documents features that have been agreed upon and may be implemented in
future Godot releases.
```