Summary:

Electrical field, or storm lightning, or sparks, everything you can imagine.

If you have some electrical effect in your game, lightning asset helps you visualize it.

Requires unity 3.5+ for runtime.
Requires unity 4.6+ for tutorials. (or just visit link).

Links:

[–=Demo project=–] [–===Tutorial===–]

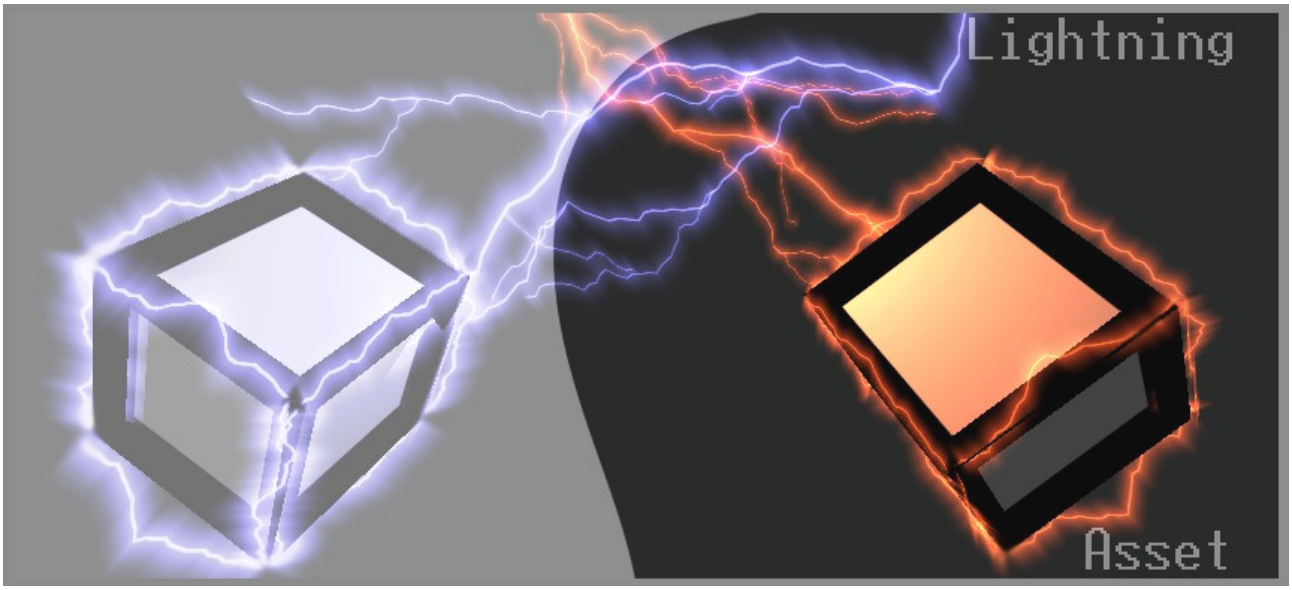
Pros:

- **Don't** need pro.
- Supports **ANY** platform.
- Easy usage. (Editor extensions, gizmos, helpers).
- Huge abilities of usage.
- Few settings, that provides a lot of capabilities.
- It's really fast.
- It's tiny. There is only one runtime library (editor and tutorials can be deleted).
- Ready to work out from box.
- No 3rd party plugins used.

Epilogue:

If something crashed, or any error appears. Please don't give me one star, or leave "this #$@! is not working" comment. This project have an Issue Tracker, please leave there a new issue, and I'll fix it. I'm just a human :-).

p.s. Thanks everybody who supports me. I try to make it better if you'll help me.



Lightning asset v1.1.0

user's manual

Table of Contents:

1. General information....4
    1.1 Versions History....4
    1.2 Package overview....4
    1.3 How it works....4
2. System requirements....5
    2.1 Unity version....5
    2.2 Operation System....5
3. Getting started....6
    3.1 Installation....6
        3.1.1 Installation from disk....6
        3.1.1 Installation from asset store....6
    3.2 User interface....6
        3.2.1 Hierarchy window menu....6
        3.2.2 Add Component menu....7
        3.2.2 Assets menu....7
    3.3 Public classes overview....8
        3.3.1 ExapleShowToPoint.cs....8
        3.3.2 ExapleShowToTarget.cs....8
        3.3.3 ExapleToggleGameObjectState.cs....8
        3.3.4 LoadSceneButton.cs....8
        3.3.5 RandomRotation.cs....8
        3.3.6 LightningObject....8
4. Using the plugin....9
    4.1 Configuring plugin....9
        4.1.1 EditorLightningSettings.asset....9
    4.2 Creating lightnings....9
        4.2.1 Create objects....9
        4.2.2 Configuring objects....9
        4.2.3 Playing....10
    4.3 Creating sparks....10
        4.3.1 Create objects....10
        4.3.2 Configuring objects....10
        4.3.3 Playing....11
5. Advanced topics....12
    5.1 Principles....12
        5.1.1 Mesh....12
    5.2 Shader....12
    5.3 Material....12
    5.4 Extensions....12
6. Conclusion....13

1. General information.

Lightning asset is unity3d package, that helps visualize electrical effects. This manual helps understand how use asset, and describes some advanced aspects of it behavior.

1.1 Versions History.

- 1.1.0 – Time update methods. Scaled Delta time, unscaled, System or Custom
- 1.0.6 – Fixed bug with DirectX shader compilation.
- 1.0.5 – Unity5.0 compatibility. Minor editor changes.
- 1.0.4 – Correct camera orientation in shader instead of set transform rotation. Some performance optimizations.
- 1.0.3 – Created EditorLightningSetting.asset file.
- 1.0.2 – Fix lightning alpha changing.
- 1.0.1 – 10% performance optimizations for sparks objects type.
- 1.0.0 – First release.

1.2 Package overview.

Folder structure:

- Example – folder that contains tutorial scenes and resources for it. (can be deleted)
- Documentation – folder that contains documentation files. (can be deleted)
- Resources – folder that contains default materials, textures, shaders.
- Scripts – folder that contains lightning libraries. (core).

1.3 How it works.

LightningObject script requires MeshRenderer and MeshFilters components. It adds those components automatically if they are not present.

After LightningObejct instantiated it initialize. During initialization it detects camera, creates mesh and assign it to MeshFilter, and other stuff. If ”Target” field configured, after instantiation it shows as like as Show() method called.

After LightningObject showed, depending on “RenderingType” field, it will:

- Fading mesh and deactivate it after lifetime. Keep word space position and rotation.

  or

- Regenerating mesh geometry. Look at camera every frame if need.

2. System requirements.

2.1 Unity version.

Recommendation:

Unity3d 4.5+ version.

Minimum:

Unity3d 3.5+ version. But you need remove “Example” folder.

2.2 Operation System.

Any windows platform.

Theoretically will works on Mac OS.

3. Getting started.

3.1 Installation.

3.1.1 Installation from disk

To install lightning package do following:

1. In Unity3d select: Asset → Import package → Custom package...
2. Select package.
3. In Import package window, press import button.

3.1.1 Installation from asset store

To install lightning package do following:

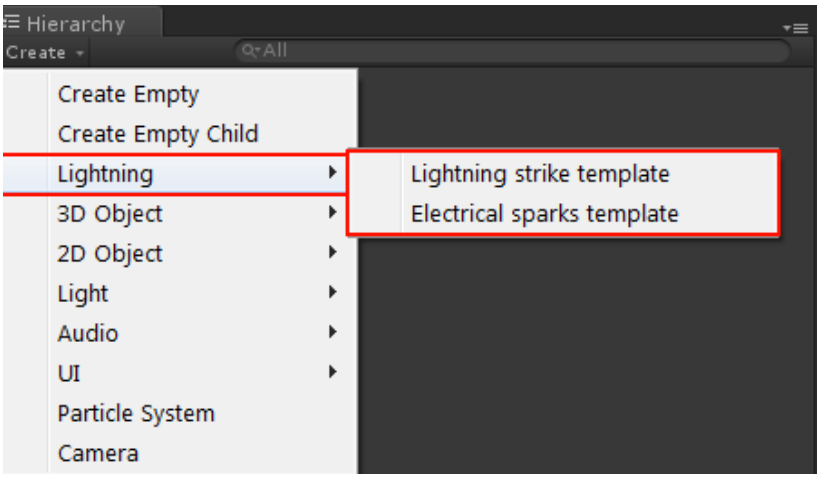
1. In Unity3d select: Windows → Asset Store. Ctrl + 9
2. Find lightning package.
3. Import.

3.2 User interface.

3.2.1 Hierarchy window menu

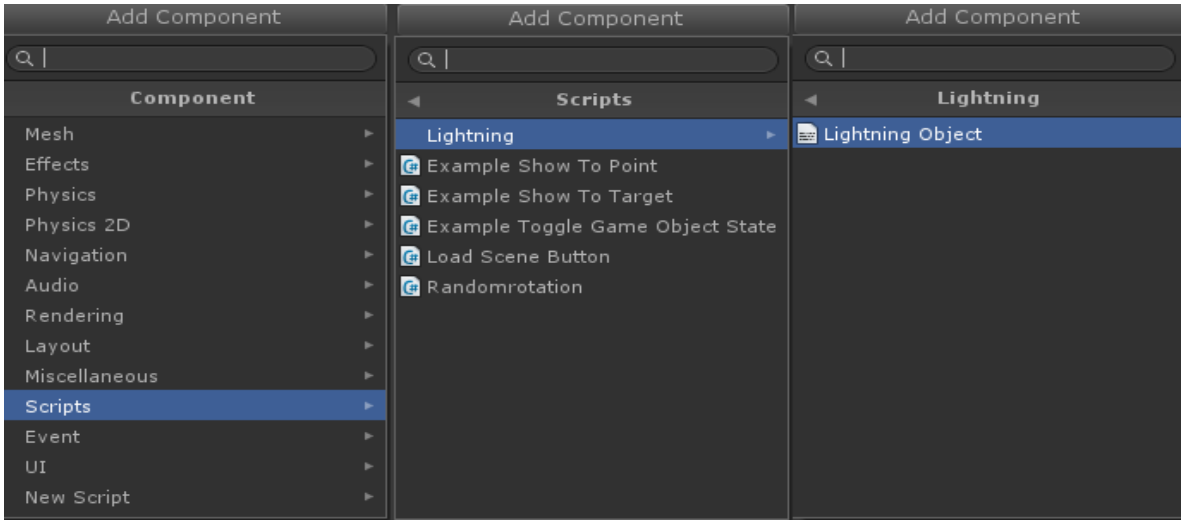
After importing package. “Hierarchy → Create” window will have new item.

- “Lightning strike template” item to create a lightning strike template object.
- “Electrical sparks template” item to create an electrical sparks template object.



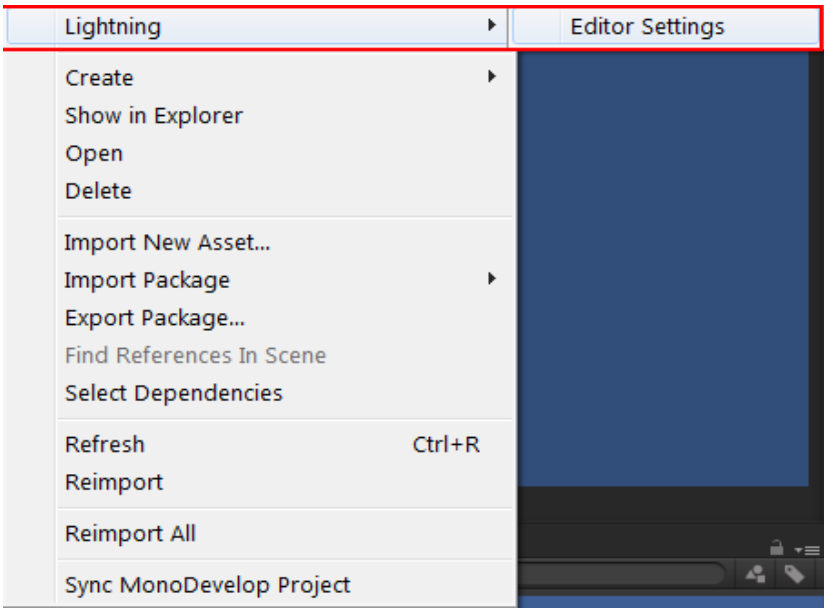
3.2.2 Add Component menu.

“Add Component” button new items.



3.2.2 Assets menu.

- “Lightning → Editor Settings” item to select or create settings asset for some editor fields. Such as default materials for different types of lightning objects.



3.3 Public classes overview

3.3.1 ExapleShowToPoint.cs

Class containing example of usage LightningObject showing to world space point as Vector3

3.3.2 ExapleShowToTarget.cs

Class containing example of usage LightningObject showing to Transform target.ctor3

3.3.3 ExapleToggleGameObjectState.cs

Class that can toggle game object “active” state.

3.3.4 LoadSceneButton.cs

Class that can load next or previous scene.

3.3.5 RandomRotation.cs

Class that rotates transform around random axis with random speed.

3.3.6 ExampleTimeScaler.cs

Class that changes time scale, and provide custom Update Time method.

3.3.6 LightningObject

Core class. Contains all logic. Have some public members.

- Show(Vector3 to); – shows LightningObject to point.
- Show(Transform to) – shows LightningObject to tarnsform.

4. Using the plugin.

4.1 Configuring plugin.

By default this package extracted to root of your project into "*Lightning*" folder. You can move it to any other location, plugin will work. After move, you should select "*Assets → Lightning → Edit Settings*" and click "*Refresh*" button. It will reset path to gizmo icon for lightning objects.

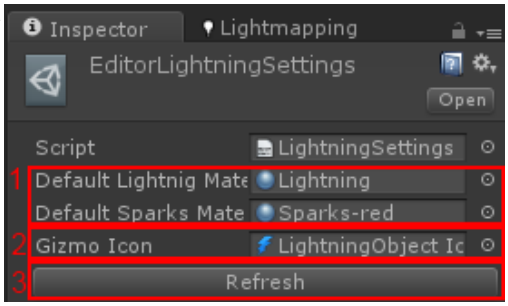
If you don't need examples – safely remove "*Example*" folder. Similarly with documentation.

"*Scripts*" and "*Resources*" folder is important folders, without them functioning impossible.

4.1.1 EditorLightningSettings.asset

Click "*Assets → Lightning → Edit Settings*" to edit plugin settings.

1. This is settings of default materials for differanr template types
2. This is setting of gizmo icon for LightningObject
3. This button will refresh gizmo icon path after (or if) you move "*EditorLightningSettings.asset"* file.



4.2 Creating lightnings.

Lighting – is object with fixed geometry (not changing during lifetime), and fading (became more and more transparent during life).

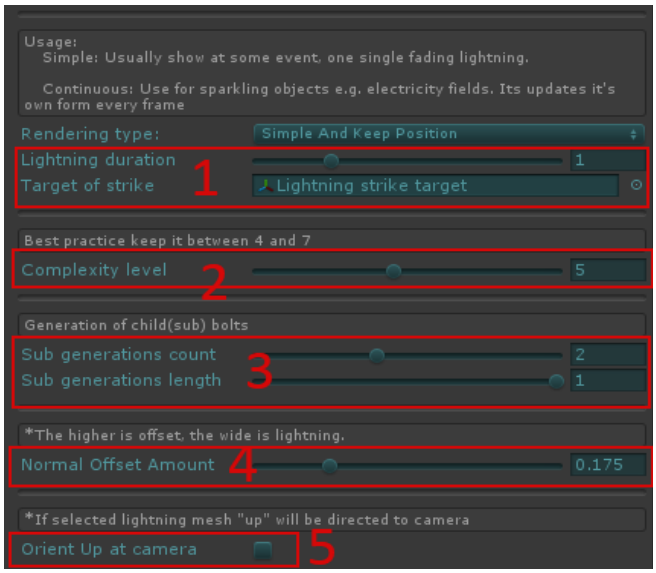
4.2.1 Create objects.

Simple way to create lightning object is using context menu in hierarchy window.

Press "Create → Lightning → Lightning Strike Template" this will create new object named "Lightning strike" with child named "Lightning strike target".

4.2.2 Configuring objects.

1. Select target of lightning and its duration.
2. Configure lightning complexity.
3. Configure children generation.
4. Configure "wideness".
5. Select if you want lightning mesh "look at" camera.



4.2.3 Playing.

There is 2 ways to show lightning:

1. Call method Show():
   - Remember local position and rotation.
   - Generate new mesh.
   - Orient mesh if camera look at orientation enabled.
   - Every frame it fades to full transparent.
   - Before deactivate returns local position and rotation.
2. gameObject.SetActive(true):
   - **NOT** remember local position and rotation.
   - **NOT** generate new mesh.
   - **NOT** orient to camera
   - Every frame it fades to full transparent.

4.3 Creating sparks.

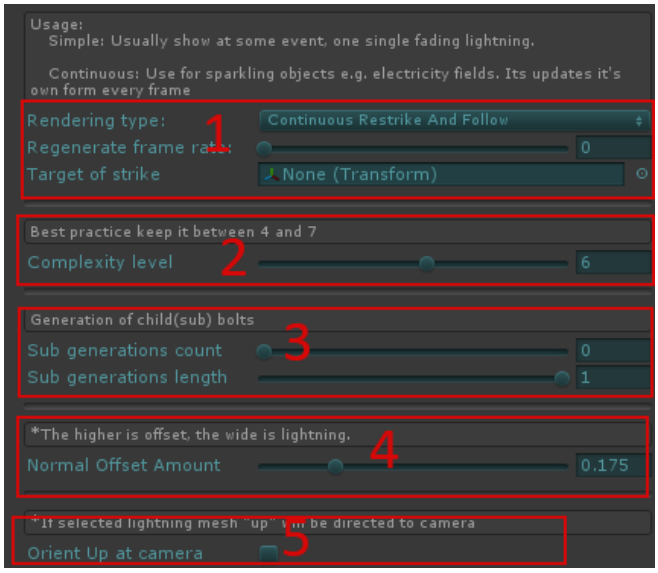
4.3.1 Create objects.

Simple way to create electrical sparks object is using context menu in hierarchy window.

Press “Create → Lightning → Electrical sparks Template” this will create new object named “Electrical sparks” with child named “Electrical sparks target”.

4.3.2 Configuring objects.

1. Select target of sparks and its regeneration rate.
2. Configure sparks complexity.
3. Configure children generation.
4. Configure “wideness”.
5. Select if you want sparks “look at” camera.



4.3.3 Playing.

There is 2 ways to show sparks:

1. Call method Show():
    - Remember target point.
    - Every frame update target point (if Target transform set)
    - Every frame it orient to camera if need.
    - Every regeneration rate: generate new mesh.
1. gameObject.SetActive(true):
    - Every frame update target point (if Target transform set)
    - Every frame it orient to camera if need.
    - Every regeneration rate: generate new mesh.

5. Advanced topics.

Principles.

5.1 Mesh.

Mesh generated as quads sequence from start point to endpoint.

Start point and endpoint transformed to LightningObject local space.

Vertex data:

- Vertex – local space vertex position.
- Normal – Vector3.right or left (shader move vertex by it).
- UV – quad uv.
- Color – Red is bolt majority width factor, Green is additional alpha (fade endpoints quads, and children start point), Blue – not used, Alpha – alpha channel.
- Tangents (v1.0.4+) – every tangent *xyz* is strike direction in object space, and *w* component indicate side of vertex: left is -0,1, right is 0,1.

5.2 Shader.

_MainTexture – Alpha texture.

_Color – Multiply color value.

_WidthScale – factor of mesh stretching to right or left depending on distance to camera.

_CameraDistanceLimit – min and max limits of stretching.

Shader created to keep mesh on screen width.

5.3 Material.

For different cases, shader properties must be configured different.

ex. for “Energy fields” not necessary keep screen width on every distance, so you can set just

5.4 Extensions.

At now, there is no ability to extend this module. And editor fields have minimum and maximum limitation. You can delete “LightnigEditor.dll” to remove limitation and set values such you want.

5.5 Time Update Method.

There are 4 types of Lightning ***currentTime*** variable update methods. Inside plugin the ***currentTime*** variable simply decrement by ***Time.deltaTime*** value. In case on ***Time.timeScale*** is set to zero, nothing happens, and plugin not worked. After version 1.1.0 there are 4 options:

1. Set *Standard Time* – default method, ***currentTime*** variable decrements by ***Time.deltaTime***
2. Set *Unscaled Time* – ***currentTime*** variable decrements by ***Time.unscaledDeltaTime***

3. Set *System* – ***currentTime*** variable decrements by value calculated inside plugin, based on ***System.DateTime***
4. Set *Custom Timer* – you can write your own class (inherited from ***Lightning.BaseLightningTimer***), that provides decrement value by overriding abstract method *GetDeltaTime()*

Note that when type set as simple – ***currentTime*** affect fading speed and whole lifetime, when type is – continuous ***currentTime*** affects regeneration rate but only if regeneration rate greater than 0. If regeneration rate set to 0, lightning would be regenerated **every** frame.

6. Conclusion.

I appreciate any help, ideas and suggestions. Please leave an issue in Issue Tracker if you have one.