Fast Line Renderer for Unity

(c) 2015 Digital Ruby, LLC

Created by Jeff Johnson

Change Log

1.2.2 (2017-02-15) - Added ScreenRadiusMultiplier to the fast line renderer script. Set to non-zero to try and keep the lines the same width on screen regardless of distance using this multiplier.

1.2.1 (2016-12-01) - Added a Mesh get/set property on the fast line renderer script. Use this to quickly clone the mesh from another fast line renderer or use it for other purposes.

1.2 (2016-09-20) - Added method to draw a grid, with or without fill

1.1.6 (2016-09-16) - Added workaround for pre-shader 3.0 where lines would not show up properly

1.1.5 (2016-06-22) - Added workaround for canvas renderer bug. Fade and lifetime is still not supported due to Unity bug.

1.1.4 (2016-05-13) - Added arrow cap to the designer scene

- Added new demo "DemoSceneList" to show how to create lines from an array of Vector3

- Added AppendCircle method to the script

1.1.3 (2016-04-27) - Fix animation speed to allow negative values and not mess up joins and caps

1.1.2 (2016-04-21) - Fix bug with splines and curves looking funny at the ends of the lines

Add animation speed properties for marching ants style animation

1.1.1 (2016-03-28) - Added animating lines to curve - spline demo and "Animation" section to this readme.txt file

1.1 (2016-03-23) - Added prefab for Unity canvas

Optimize shader by removing if - else conditional

Added ChangePosition method to move line positions

Resource leak cleanup

1.0.1 (2016-03-15) - Line renderer takes the layer of the parent game object

- Fix shader on mobile

1.0 (2016-01-21) - Initial release

Please read ALL of the following to get the most out of this asset. By reading this documentation, you will avoid a lot of problems and frustrations.

Make sure to include "using DigitalRuby.FastLineRenderer" in your scripts.

Fast Line Renderer is a very powerful asset for rendering many quads and lines in Unity. With a fantastic shader that leverages the GPU, your CPU will be free to do more important tasks.

If you are wanting the lines to stay the same radius regardless of distance, try out the ScreenRadiusMultiplier on the script.

Use Cases

--------------------------------------------------------------------------------

- Performance critical particle systems - Fast Line Renderer runs entirely on the GPU. Once the lines are created, the CPU work is done. Billboarding, jitter, turbulence, velocity, rotation and fade in/out are all GPU driven. Multiple distinct line segments are possible in a single mesh. In contrast, the Unity Line Renderer runs mostly on the CPU and requires one line per object.

- Designer / inspector creation - Create your lines right in the Unity inspector to immediately visualize them without having to run your scene.

- Soft lines - The Fast Line Renderer shader has a soft particles option. When your lines intersect solid geometry, they will look nice.

- Grids - Fast Line Renderer can easily draw grids, with or without fill for 3D cubes.

- GUI - Fast Line Renderer works great for a custom, performant GUI as it renders entirely on the GPU.

- \*Note- There are a few Canvas bugs that limit Fast Line Renderer on a Unity Canvas. I have notified Unity of these bugs and they say they hope to have them fixed for Unity 5.4. Namely glow doesn't work, and the lines disappear.

- More - your imagination is the only limit!

Folders

--------------------------------------------------------------------------------

- Demo: Contains demo scenes and script

- Material: Premade material for lines with and without glow, along with start and end caps

- Prefab: Premade game object with fast line renderer script ready to render lines with glow and start and end caps

- Script: Fast line renderer script and path generator script

- Textures: Line and glow textures with caps

Demo

--------------------------------------------------------------------------------

- DemoSceneCanvas: Demonstrates fast line renderer on a Unity canvas. There are a few bugs that I've notified Unity about and they say they will be fixed in Unity 5.4. Namely glow doesn't work, and the lines disappear.

- DemoSceneCurves: Demonstrates curves and splines

- DemoSceneDesigner: Shows how to create lines in the inspector

- DemoSceneEffects: Example of a simple fireball effect

- DemoSceneGrid: Shows how to draw grids, see the DoGrid method

- DemoScenePerformance: Demonstrates performance vs. the Unity Line Renderer. Watch as the FPS drop quickly with the Unity Line Renderer.

- DemoSceneScript: Shows how to create temporary lines (like a particle system) in script.

Prefab

--------------------------------------------------------------------------------

I've setup some prefabs for you to drag right into your scene.

- FastLineRenderer: Prefab with glow setup ready to go

- FastLineRendererCanvas: Prefab with glow setup ready to go on a Unity Canvas

- FastLineRendererCurveSpline: Prefab setup with glow and will look nice with curves and splines

- FastLineRendererGrid: Prefab that is setup to draw grids nicely

- FastLineRendererGridNoGlow: Prefab that is setup to draw grids nicely, without glow option for performance

- FastLineRendererNoGlow: Prefab setup to render lines that don't need glow, curves or splines

Script

--------------------------------------------------------------------------------

FastLineRenderer script is the work horse of this asset. The script has many configuration options. Only one script per game object is supported. Multiple fast line renderer scripts on one game object is not supported.

The simplest use case is creating a line from a list of points. See FastLineRendererDemoList.cs for an example in the scene DemoSceneList.

- Material: The material the lines will use to render with

- Clone Material: Whether to clone or share the material. Almost always you will want this checked, unless you are trying to optimize performance for fewer draw calls. When this is set to false, the material is shared so any modifications made (even at runtime) will apply to all material, including the assets folder.

- Camera: The camera that the script will be rendering in. Defaults to the main camera.

- AnimationSpeed: Offsets the uv of the line to do marching ants style animations.

- TintColor: Apply a tint to the line texture.

- StartCapScale: Scale just the start caps.

- EndCapScale: Scale just the end caps.

- Line UVX Scale: Typically this value is left at 1 unless you want to render a dotted line or other line with a repeating pattern. In this case, ensure your texture is set to REPEAT instead of CLAMP.

- Line UVY Scale: Almost always this will be 1 unless your texture has a vertically repeating pattern (rare).

- Glow UVX Scale: Same as line except it applies to the glow.

- Glow UVY Scale: Same as line except it applies to the glow.

- Glow Color: Applies a tint to the glow texture.

- Glow Intensity Multiplier: Increases the final alpha value of the glow.

- Glow Width Multiplier: Applies a multiplier to each glow width.

- Glow Length Multiplier: Applies a multiplier to each glow segment.

- Jitter Multiplier: Causes the line to distort, entirely on the GPU. Great for cartoon effects.

- Turbulence: Causes the lines to move in the direction they are pointing, entirely on the GPU.

- Bounds Scale: Multiply the bounds of each mesh created to ensure that the Unity culling works. For lines with jitter, turbulence, rotation and velocity you may need to set this higher than 1 since the GPU is moving the lines and the mesh remains the same.

- Line Texture: Texture to use for each line.

- Line Texture Start Cap: Texture to use for start caps.

- Line Texture End Cap: Texture to use for end caps.

- Line Texture Round Join: Texture to use for round joins (typically a circle).

- Glow Texture: Texture to use for glow.

- Glow Texture Start Cap: Texture to use for start glow.

- Glow Texture End Cap: Texture to use for end glow.

- Glow Texture Round Join: Texture to use for round join (usually null).

- ScreenRadiusMultiplier: Experimental, attempts to keep lines the same width on screen using this multiplier.

- Initial Line Groups: Covered in the next section, "Designer View".

- Sort Layer Name: For 2D, allows changing the sorting layer.

- Sort Order In Layer: For 2D, allows changing the order within the sorting layer.

Moving Line Positions: Call ChangePosition on the FastLineRenderer script. Call Apply once all your changes have been made. Line joins and caps are not supported with this method, for those simply recreate all your lines from scratch using the reset and apply methods.

Designer View

--------------------------------------------------------------------------------

Fast Line Renderer supports creating your lines in the Unity inspector. This is great for static content that you want to create visually. See DemoSceneDesigner for a full demo.

The script contains an "Initial Line Groups" property. Each group represents a set of lines with the following properties:

- Description: Create a name for the group to help you remember what it represents

- Offset: Offset the group of lines by x, y and z amount

- Line Radius: Radius of each line in world units

- Line Color: Color of each line

- Glow Width Multiplier: Each line can have a different glow width. Ensure you are using a material that supports glow.

- Glow Intensity: Each line can have a different glow intensity

- Add Start Cap: Whether to add a start cap to the first line. Ensure that your material supports caps.

- Add End Cap: Whether to add an end cap to the last line.

- Line Join: Type of line join.

- Continuous: Whether the line is one long segment or individual lines. If true, then each point appends a line. If false, every two points represents an individual line and the line join is ignored.

- Points: The points in the line. For continous lines, each point appends a new line. For non-continous lines, every two points represents an individual line.

Animation

--------------------------------------------------------------------------------

Animating a line is really easy. The FastLineRendererProperties class contains an AddCreationTimeSeconds method that delays creation of line segments. Simply call this method on your properties as you add line segments, and they will animate in.

Curves and splines have an animation time parameter built in that is cumulative per line segment.

Marching ants animation is also easy. If you look in the designer demo scene, there is a dotted line object under the Lines object. AnimationSpeed controls how fast the dots move and LineUVXScale controls how many dots there are per line segment. There are also matching glow properties for animation speed and uv x scale. Ensure your texture is set to wrap mode = repeat for this animation.

Line Joins

--------------------------------------------------------------------------------

Fast Line Renderer has several different join modes, accessible on the properties object (FastLineRendererProperties).

- None: Ideal for opaque, thin lines as there is no CPU or GPU overhead

- AdjustPosition: Move each line segment back a little bit so that it intersects with the previous line. Great for opaque lines that are not very thick.

- AttachToPrevious: The best choice for translucent or 3D lines that move in both x, y and z directions. Also the preferred choice for curves and splines. Each line attaches to the two end vertices of the previous line.

- Round: Great for opaque, thick lines as the join looks nice and smooth.

Line Caps

--------------------------------------------------------------------------------

- Start: Add a start cap. Only useful when starting a line.

- End: Add an end cap. Only useful when ending a line.

- None: No cap. Use this for anything other than a start or end segment.

Note: All line segments, caps and round joins are rendererd on the GPU in a single draw call, so there is no overhead or performance impact.

Start and end cap may be scaled via script parameters, that way arrow caps or other types of caps can be added that are wider or longer than a normal line segment. Simply tweak your start and end cap scale until they look right, then as you change the line radius, you shouldn't need to change the start and end cap scale.

Material

--------------------------------------------------------------------------------

Fast Line Renderer contains two separate materials.

- FastLineRendererMaterial: Material setup with glow. Renders in two passes.

- FastLineRendererMaterialNoGlow: Material setup with no glow. Renders in one pass.

Shader

--------------------------------------------------------------------------------

Fast Line Renderer contains two shaders that match up with the material (glow and no glow). FastLineRendererShader is the base shader.

The shader is an advanced, innovative shader that allows rendering complex effects such as fade in/out, rotation, jitter, turbulence, velocity and billboarding all on the GPU.

Start caps, end caps, line segments and round joins are all done in a single draw call to improve performance.

For extra performance:

- If you aren't using glow, make sure to use the no glow material and shader.

- If you aren't using round joins, start and end caps, enable the keyword "DISABLE\_CAPS" on the material to reduce texture lookups.

Creating lines in script

--------------------------------------------------------------------------------

To use the line renderer in script, typically you will call CreateWithParent, passing in a template FastLineRenderer, i.e. FastLineRenderer r = FastLineRenderer.CreateWithParent(null, LineRenderer);

Once you have created your Fast Line Renderer, you can create lines in script using the following methods:

- AddLine: Add a distinct single part line segment with no caps or join.

- StartLine: Start a new line segment.

- AppendLine: Append a line to the previous position in the current line segment.

- EndLine: Finish a line segment.

- AppendCurve: Add a curve which smoothly curves using a start point, end point and two control points.

- AppendSpline: Add a spline with a start point and end point that curves smoothly through 4 or more control points.

Each of these methods takes a FastLineRendererProperties object which contains details about how the line should behave. These properties are applied per line segment rather than per line group, allowing for some interesting effects.

- Start: The start point of the line. For some methods, this is all that is required since the line will be appended.

- End: The end point of the line. For some methods, such as AppendLine, this is ignored.

- Radius: The radius of the line in world units.

- Color: A tint color to apply to the line.

- GlowWidthMultiplier: Apply additional glow width if desired.

- GlowIntensityMultiplier: Apply a dimmer or brighter glow if desired.

- LifeTime: Do not modify this, instead call SetLifeTime on the properties to set the creation time, fade in/out seconds and life time in seconds.

- Velocity: Velocity of the line in world units per second.

- Angular Velocity: Angular velocity of the line in radians per second.

- LineJoin: The join mode if the line is being appended.

- LineType: The line type. Normally this is set to full.

Once you have created all of your lines, call Apply to make the changes permanent. See the UpdateDynamicLines in the demo script for an example.

For Curve and Spline, see the AddCurvesAndSpline method in the demo script.

If your lines are temporary, you should free up your Fast Line Renderer after the maximum life time, i.e. r.SendToCacheAfter(TimeSpan.FromSeconds(maxLifeTimeSeconds));

To create quads, simply specify a start and end position that is a distance equal to the radius \* 2.

Caching and Dynamic Lines

--------------------------------------------------------------------------------

FastLineRenderer has a built in caching system to aid with advanced scenarios with particle systems and changing lines.

- CreateWithParent: Use this static method to quickly create FastLineRenderer objects. For dynamic lines that change frequently, simply request a new FastLineRenderer and add the appropriate lines.

- SendToCacheAfter: Use this method to send FastLineRenderer objects back to the cache after a certain amount of seconds. Great for when you are creating temporary effects.

- Reset: Use this method to remove all lines immediately.

2D mode

--------------------------------------------------------------------------------

Fast Line Renderer has optimizations for 2D mode and ignores the z coordinate. If your camera is set to orthographic, 2D mode is enabled automatically in the shader.

5 million lines at 60 FPS

--------------------------------------------------------------------------------

Depending on your CPU and GPU, your performance may vary. I tested on my laptop which has an Intel Core i7-6820HK and Geforce 980M. Quality settings, soft particles and camera render modes may affect performance.

This demo also uses the Mesh property of the script to quickly create additional fast line renderers.

Canvas

--------------------------------------------------------------------------------

If using in Canvas, make sure to set additional shader channels to everything in the inspector

Limitations

--------------------------------------------------------------------------------

Fade and lifetime are not supported on DX9 and shader model < 2 due to the limited amount of vertex information that can be passed.

Feel free to email me (support@digitalruby.com) if you have questions.

- Jeff