Riemann’s Lens Manual

Aaron M. Schinder

7 May 2022

(This manual is still in development)

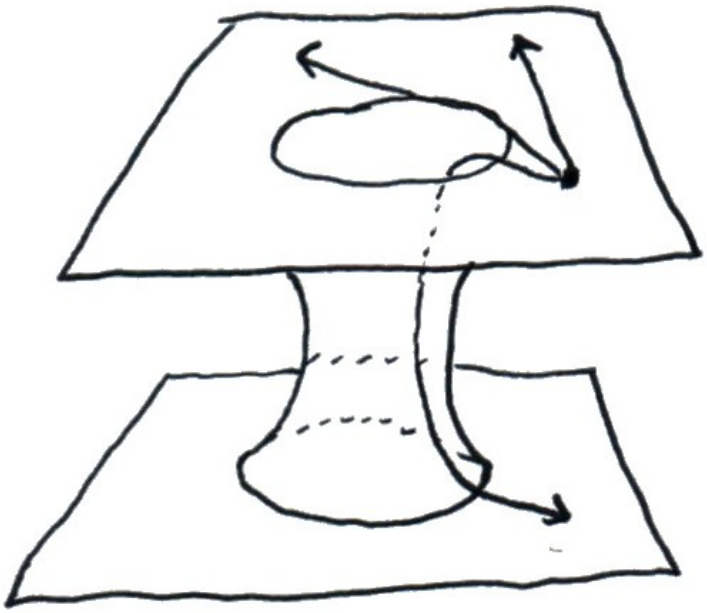
Introduction

Wormholes are a staple of science fiction. They are strange topological features of distorted space which can connect otherwise distant regions of space and time with a bridge of a shorter length, or perhaps connect one space with a different space entirely, connected in no other fashion to the original.

Unlike the more familiar black holes, which we have found in astronomy, a wormhole notionally allows you to end up somewhere more interesting than an infinitesimal point, and can potentially be traversed reversibly.

In most television science fiction, they are depicted as a glowy tunnel, maybe with something swirly going on on the 2-d circular entry and exit-points. :-/ Then again, television science fiction also depicts orbital dog-fighters dropping gravity bombs, so ....

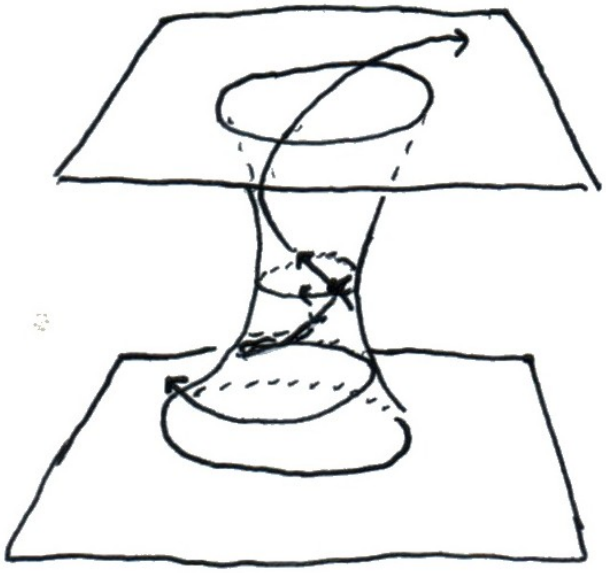
Anyway, if you think about it, the question of what a wormhole would end up looking like, and what traversing one would end up being like is an interesting one, and this simulation is intended to provide one to you to play with.



The usual illustration of a wormhole is of a two-dimensional curved surface in a higher three-

dimensional “embedding euclidean space”. The upper and lower portions are flat 2-d regions, and they are joined by a kind of tunnel. However, it takes the metaphor too literally to think that this is what would appear to a “flatland ant” living in the 2d manifold, or to us living in our 3d space if we were to encounter a 3d wormhole.

The flatland ant, native to the 2d manifold would see the world in terms of light rays that propagate on the curved surface of the manifold. The wormhole itself is a feature of the space – it would not appear directly – only matter within the space interacts with the light.



The “tunnel wall” isn’t a wall to the flatland ant, it’s the bridge over which it can see and travel.

If the ant were in the neck of the wormhole, every direction in which a ray could be sent would end up going somewhere, to either end of the wormhole, with one exception: There would be a limiting direction which would travel around the neck and end up back at the start.

Control Keys

F1	Display in-game interface help
F4	Toggle simulation state HUD
Keypad 1,2,3	Cycle scenarios. Keypad 3 will attempt to load the custom scenario script located in ../data/customscenario.txt
Arrows, Mouse button 1	Rotate View
W,S,A,D	Translate view, move forward/backward/strafe left/right

Q,E	Roll left/right
H	Increase wormhole outer radius
J	Increase wormhole inner radius
K	Increase wormhole length
B	Decrease wormhole outer radius
N	Decrease wormhole inner radius
M	Decrease wormhole length
Keypad 7	Use textured shader
Keypad 8	Use two-color debug shader (I kept it in because it produces a neat stainglass look)
[,] (bracket keys)	Increase/Decrease field-of-view angle (zoom in/out)
Keypad +	Increase render resolution by factor of sqrt(2): Warning! If you increase this too much, the program may hang.
Keypad -	Decrease render resolution: Useful if you are positioning yourself within a complicated scene that might render too slowly.

Custom Scene Scripting Language

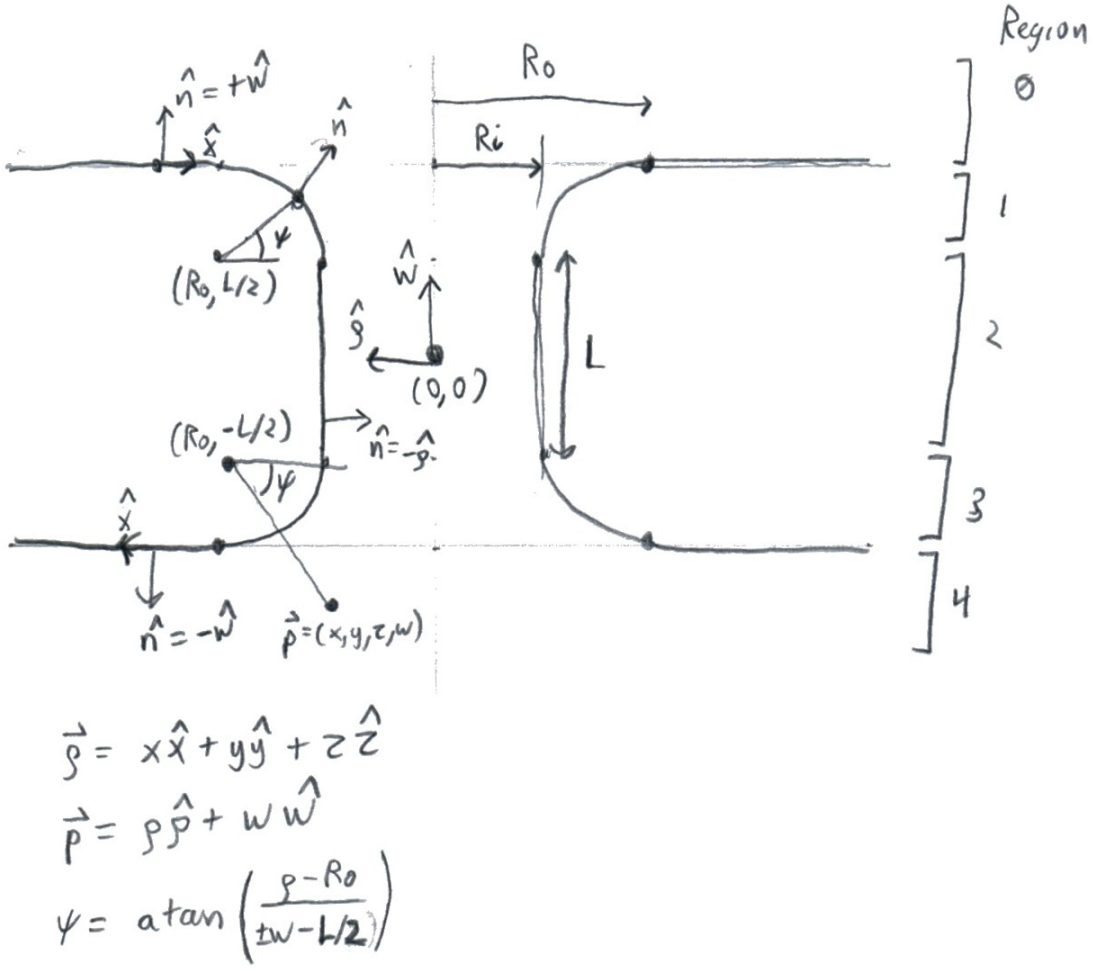
To make this simulation more interesting and useful, I wanted to provide a way to create your own scenery on either end of the wormhole. The keypad key 3 loads a custom scenario script located in the application folder: **./data/customscenario.txt** is parsed to construct the scene.

The scripting language is composed of whitespace-seperated tokens terminated by semicolons. Other additional whitespace is ignored. Comments are C style.

Command	**Parameters and Description**
begin_xxx/end_xxx	Specific blocks (viewpoint, wormhole, object, realm, etc) inside of which the attributes of the particular object are being defined
viewpoint:	
position	x,y,z,w: Initial position of the viewpoint
orientation	4x4 orientation matrix (xyzw)x(xyzw) for 4d euclidean embedding space
rotate4d	Alternative way to construct an orientation: Format is manifold normal (xyzw), rotation axis (xyzw) and angle
fovx, fovy	Angle in degrees.
wormhole:	Wormhole definition block
Router	Wormhole starting outer radius
Rinner	Wormhole starting inner radius
Length	Wormhole neck length

smax	Max ray-marching length to find one euclidean end of the wormhole before terminating. Decrease to improve performance.
ds	Ray-marching step length through the noneuclidean space
light_ambient	(r, g, b, magnitude): ambient light
light:	Positioned point-source light blocks
position	x,y,z light position
color	r,g,b,magnitude
magnitude	Additional distance-based attenuation is magnitude/(1.0+linear*dist+quad*dist^2)
linear_atten	
quadratic_atten	
object:	Object blocks are bags of mesh blocks that live in one of two Euclidean realms on either end of the wormhole
translate	(x,y,z) translate the whole object
rotate	(x,y,z,angle) rotation axis and angle (degrees)
mesh:	Mesh blocks should be defined within object blocks
Specific mesh type initializers	
quad	Rectangle with texturing applied either pinned to the corners, or repeated over the surface with a given lengthscale.
sphere	Sphere with equirectangular texture coordinates applied. Nlat, Nlon, radius, center
torus	Torus with texture coordinates applied. Nphi, Ntheta, Rmajor, Rminor
general	A general triangle mesh with vertices, normals, and texcoords specified with the vertices, normals, and texcoords sub-commands.
General mesh parameters:	
color	r,g,b,a: diffuse color
specular_color	Color for specular phong-reflection
phong_angle	Angle of the reflected lobe
emissive_color	Color for the surface emitted independently of other light sources
texture	The name of a texture that is loaded using the load_texture command. Defaults to “”, which is an untextured object
load_texture	load_texture “filename” as “internal_name”: Attempts to read a texture in an image file (png, jpg, bmp).

Geometry of This Wormhole



Rather than attempting to solve the geodesic problem within a series of maps defined in 3-d space, this simulation “cheats” and models the noneuclidean manifold directly as a 3-d shape in a higher four dimensional euclidean embedding space. Points in the space are given by four coordinates, x,y,z,w, and are constrained to the manifold as needed.

The wormhole depicted in this simulation is composed of smooth piecewise-continuous sections joined together. Regions 0 and 4 are the euclidean regions on either end of the wormhole (realms 0 and 1 in the scripting language.) In these regions, space is flat, the manifold has manifold normals pointing in either the +w or -w directions, and vectors within the manifold are spanned by the x,y,z vectors.

When the radius rho drops below the outer radius of the wormhole, the viewpoint enters the noneuclidean portion of the manifold. Two spherical-frustrum sections are joined by a spherical-cylinder neck. In the neck the manifold normal points in the -rho direction, and the manifold tangent space is spanned by w and local vectors normal to -rho.