Laboratório VISGRAF
Instituto de Matemática Pura e Aplicada

Visualization of Non-Euclidean Spaces using Ray Tracing

Luiz Velho, Tiago Novello, Vinicius Silva, Djalma Lucio

Technical Report TR-19-09 Relatório Técnico

September - 2019 - Setembro

Visualization of Non-Euclidean Spaces using Ray Tracing

Luiz Velho
VISGRAF Laboratory
Rio de Janeiro, Brazil
lvelho@impa.br

Vinicius da Silva
VISGRAF Laboratory
Rio de Janeiro, Brazil
dsilva.vinicius@gmail.com

Tiago Novello
VISGRAF Laboratory
Rio de Janeiro, Brazil
tiago.novello90@gmail.com

Djalma Lucio
VISGRAF Laboratory
Rio de Janeiro, Brazil
dlucio@impa.br

***Abstract*—This paper presents a system for immersive visualization of Non-Euclidean spaces using real-time ray tracing. It exploits the capabilities of the new generation of GPU's based on the NVIDIA's Turing architecture in order to develop new methods for intuitive exploration of landscapes featuring non-trivial geometry and topology in virtual reality.**

I. Introduction

In late 2018, NVIDIA introduced a new generation of GPUs that according to Jensen Huang, the company's CEO, is a major breakthrough in the history of the computer graphics industry. The Turing architecture of the RTX GPUs was developed over the past 15 years to make possible the implementation of ray tracing algorithms in real-time, thus enabling visualization applications with an unprecedented degree of photo-realism.

In this paper, we take the challenge of applying the power of this new generation of RTX GPUs in the exploration of mathematical spaces that feature non-trivial geometry and topology in virtual reality.

A. Motivation

The Turing architecture combines traditional capabilities of the previous generations GPUs for *rasterization* (Graphics Pipeline) and *compute* (CUDA) with new capabilities for *artificial intelligence* (Tensor Core) and *ray tracing* (RT Core). Overall, these features in aggregate form a powerful set of complementary resources for the development of new media applications, not possible before.

While the most obvious use of the RTX GPUs is for real-time photo-realistic simulation with applications in Entertainment, Architecture, Design and etc., there are other areas where this power can open up new perspectives not imaginable before. One of these areas is the *Visualization of Mathematics*. In this realm, abstract concepts, such as: High Dimensional Spaces; Non-Euclidean Geometries; Non-Trivial Topologies and Manifolds, can be made concrete for immersive exploration.

Using Virtual Reality and Ray Tracing it is now possible to create these Mathematical landscapes for interactive visualization, literally putting the viewer inside these abstract worlds for an intuitive understanding. Such experiences have the potential to allow many insights with great impact in research and education, among other aspects.

B. Contributions

The main contribution of our work is the development of an experimental platform for the immersive visualization of Non-Euclidean Spaces using real-time ray tracing.

This includes the design and implementation of an extensive framework for creating interactive experiences in landscapes that can model different types of three-dimensional Manifolds/Orbifolds.

In addition, in order to test our platform and validate its effectiveness, we produced a series of virtual reality applications and conducted informal user studies that gives directions for future research.

The system is implemented on top of NVIDIA's Falcor [1] real-time rendering framework using DirectX 12 (DXR) on Windows 10. For that, we had to extend Falcor in order to integrate Ray Tracing with Virtual Reality [2].

C. Structure of the Paper

The paper is structured as follows: Section II reviews previous and related work; Section III introduces the basic mathematical concepts associated with our work; Section IV presents the method for GPU ray tracing of 3D Manifolds/Orbifolds; Section V shows examples of experiments and discusses the analysis of our results; Section VI elaborates on possible extensions of our platform and suggests perspectives for future work; finally Section VII provides concluding remarks.

II. Related and Previous Work

In this section we review previous work for visualization of Non-Euclidean spaces and report on other related work that are relevant to our research.

A. OpenGL Visualization of Non-Euclidean Spaces

Historically, the main effort for mathematics visualization, particularly of Non-Euclidean Spaces, took place at the Geometry Center during the period of 1994 to 1998. This initiative, under the leadership of William Thurston, resulted in a scientific program to study and disseminate modern geometry using interactive visualization. Since Thurston's personal research focused primarily on hyperbolic manifolds [3], it was natural that the Geometry Center investigated the visualization of manifolds and orbifolds.

For this purpose, a platform called *Geomview* [4] was developed. The software was based on OpenGL and supported interactive viewing in Euclidean, elliptical and hyperbolic geometries.

Geomview featured a plugin architecture that made possible, among other things, the development of a module for the visualization of manifolds [5].

B. Virtual Reality

Researchers at the Geometry Center, already at that time realized the potential of Virtual Reality for providing insights into the world of geometric structures. Therefore, they created simple VR installations to allow the user, not only to have a glimpse at the visual landscape inside a 3-manifold, but also to experience the sensation of being immersed in such an environment. Two of their projects are Mathenautics [6] and Alice [7].

Another initiative in the direction of using virtual reality for mathematics visualization was JReality [8], a Java based 3D scene graph package designed for mathematical visualization at TU-Berlin. It can be used for creating immersive views of 3-manifold and relies on JOGL as a back-end for interactive OpenGL rendering.

C. Ray-Tracing

The early work on interactive visualization of Non-Euclidean spaces, reported above, was based on the traditional OpenGL rasterization pipeline. Therefore, the rendering algorithms employed a *Scene-Based* architecture.

The first work to propose the use of an *Image-Based* architecture for visualization of Non-Euclidean spaces using GPUs was from Berger et al. [9]. Their rendering algorithm exploited programmable compute shaders and CUDA to implement ray-tracing on the GPU.

Currently, the NVIDIA's Falcor rendering framework [1] provides a platform based on Vulkan an DirectX 12 that supports many features for real-time visualization, including OpenVR and DirectX Raytracing. However, as we mentioned above, ray tracing and virtual reality do not work in an integrated way in Falcor.

D. Metric Neutral

One relevant aspect in the visualization of Non-Euclidean spaces is the metric implied by the geometry. In this respect, Gunn [10] proposed a metric-neutral framework that simplifies the rendering of such geometries. Particularly, it introduces some advances that have an impact on the architecture of generic Virtual Reality systems — for example, a metric-neutral algorithm for head-tracking in VR for the different metric spaces of interest.

Another approach that simplifies the implementation of rendering applications for Non-Euclidean geometries was proposed by Guimaraes et al. [11]. It is an encapsulation method to dissociate the application development from the geometric space in which it will be represented, while at the same time preserving the intrinsic metric and topological structures of the space.

III. Basic Concepts

In this section we define the concepts and main results of *manifolds* and some special *non-manifolds*: *polyhedral complexes* and *orbifolds* for combinatorial and algebraic contexts, respectively. We also present the ingredients for a ray tracing implementation on such abstract spaces.

A. Manifolds

The paper deals with an immersive visualization of Non–Euclidean geometries using ray tracing, thus at least three properties are required in such spaces:

- Being locally similar to an Euclidean space. This allows us to put the viewer and the scene inside the ambient as in the common approaches;
- For each point p we need vectors pointing in all directions: the *tangent vectors* in p. Moreover, the *inner product* between two tangent vector is necessary. These definitions are used to simulate effects produced between the lights and the scene objects.
- Finally, for a point p and a vector v "tangent" in p, we should be able to compute the *ray* leaving p in the direction of v.

Riemannian manifolds are example of space satisfying the above properties and they are described below. Lee [12] is a good reference for manifolds. We start with Euclidean spaces, where ray tracing is commonly used.

Example 1 (Euclidean space)**.** *The 3-dimensional Euclidean space $\mathbb{E}^3$ is the set $\{(x, y, z) |\ x, y, z \in \mathbb{R}\}$ endowed with the classical inner product $\langle u, v\rangle = u_x \cdot v_x + u_y \cdot v_y + u_z \cdot v_z$ where $u = (u_x, u_y, u_z)$ and $v = (v_x, v_y, v_z)$ are vectors in $\mathbb{E}^3$. The distance between two points p and q is defined by $d_{\mathbb{E}}(p, q) = \sqrt{\langle p - q, p - q\rangle}$. The curve $\gamma(t) = p + t \cdot v$ describes a ray leaving a point p in a direction v. Analogously, for a $n > 0$ the n-dimensional Euclidean space $\mathbb{E}^n$ is constructed.*

A *n-manifold* M is a topological space which is locally identical (topologically speaking) to the Euclidean space $\mathbb{E}^n$; n is the dimension of M. More precisely, there is a neighborhood of every point in M mapped homeomorphically to the open ball of $\mathbb{E}^n$. These maps are called *charts* of M. The change of charts between two neighborhoods in M must be continuous. Thus, informally, the manifold definition generalizes the concept of Euclidean spaces. This work focus on manifolds of dimension 3. Examples of 3-manifolds include the unity sphere in $\mathbb{E}^4$ and the torus $\mathbb{T}^3 = \mathbb{E}^3/\mathbb{Z}^3$.

Straight lines are fundamental objects when working with ray tracing algorithms, since light travels along them. A manifold M admits a generalization of such notion, the *geodesics*. To define them we need two additional tools. The first is the calculus framework, which is done by requesting that changes of charts in M are diffeomophisms — M is called *differentiable*. This allows us to define for each point a *tangent space* and work with calculus on it. The second tool is the attribution of a appropriate metric on each tangent space — M is called *Riemannian*. Then we can compute angles between

vectors in tangent spaces (crucial in ray tracing), and distances between two points in M. Finally, a *geodesic* in M is a curve such that locally it is the shortest path. We use the term *ray* instead of geodesics since the paper deals with ray tracing.

Two classical Non-Euclidean 3-manifolds (elliptic and hyperbolic) are described below.

Example 2 (Sphere)**.** *The unity 3-sphere $\mathbb{S}^3$ inside $\mathbb{E}^4$ is the set $\{p \in \mathbb{E}^4 |\ d_{\mathbb{E}}(p,0) = 1\}$ endowed with the induced Euclidean inner product on each tangent space. The distance between two points p and q in $\mathbb{S}^3$ is defined by $d_{\mathbb{S}^3}(p,q) = \cos^{-1}\langle p, q\rangle$ where p and q are considered vectors in $\mathbb{E}^4$.*

A ray in $\mathbb{S}^3$ passing through a point p in a tangent direction v is the arc produced by the intersection between $\mathbb{S}^3$ and the plane spanned by v and p of $\mathbb{E}^4$.

The 3-sphere $\mathbb{S}^3$ is an example of a Non-Euclidean geometry, since its fails the Parallel Postulate: given a ray $\mathcal{R}$ and a point $p \notin \mathcal{R}$, there is a unique ray parallel to $\mathcal{R}$. As the rays in $\mathbb{S}^3$ are the big circles, thus choosing two of then in $\mathbb{S}^2 \subset \mathbb{S}^3$, they intersect in exactly two points.

Example 3 (Hyperbolic space)**.** *The 3-hyperbolic space $\mathbb{H}^3$ is the hyperboloid $\{(w,p) \in \mathbb{E}^+ \times \mathbb{E}^3 |\ w^2 - \langle p,p\rangle = 1\} \subset \mathbb{E}^4$ endowed with a special metric*

$$d_{\mathbb{H}^3}((w_1,p_1),(w_2,p_2)) = \cosh^{-1}(w_1 \cdot w_2 - \langle p_1, p_2\rangle)$$

where (w_1,p_1) and (w_2,p_2) are two points in $\mathbb{H}^3$.

Rays in $\mathbb{H}^3$ are the intersections between $\mathbb{H}^3$ and the planes in $\mathbb{E}^4$ containing the origin. For instance, the ray leaving a point $p \in \mathbb{H}^3$ in a direction v is the intersection between $\mathbb{H}^3$ and the plane spanned by the vectors v and p in $\mathbb{E}^4$.

The space $\mathbb{H}^3$ does not contain any straight line, thus its rays can not be straight. As in the sphere case, $\mathbb{H}^3$ is a Non-Euclidean space, since its does not satisfies the Parallel Postulate. For a ray $\mathcal{R}$ and a point $p \notin \mathcal{R}$ there are many rays passing through p which do not intersect $\mathcal{R}$.

Hopefully, there are many combinatorial and algebraic constructions of manifolds which allow us to represent such exotic structures in computers. We focus on two notions: identifying faces of a polyhedra and quotient by discrete groups. Both constructions depart from a purely topological point of view, so we add a geometry, which we consider to be Euclidean (other geometries are left to future works).

B. Manifolds as the identification of polyhedra faces

We start with a combinatorial way to build manifolds. Consider first the two dimensional case. Let there be given a finite collection of convex polygons with their edges divided into disjoint pairs. Identifying each couple of edges through a homeomorphism gives rise to space K. The well-known classification theorem of compact 2-manifolds states that K is a manifold. Triangulations and quadrangulation of compact surfaces are common examples of such structures.

There is an analogous procedure for the three dimensional case. Let a finite number of (convex) polyhedra be given. Endow such polyhedra with an appropriate pair-wise identification of its faces. Each couple of faces has the same number of edges and are mapped homeomorphically to each other. Such identification gives rise to a particular *polyhedral complex* K, which is not necessary a 3-manifold. However, K is a 3-manifold if, additionally, its Euler characteristic is equal to zero (Theorem 4.3 in [13]).

In this paper we only consider the polyhedra (used above) embedded in $\mathbb{E}^3$, this induces a geometry on K which is Euclidean (flat) restricted to each polyhedron. Thus the scene (for the ray tracing) can be created in the classical way: no geometry distortion is required. The spherical and hyperbolic cases are future work.

Probably the most famous example provided by the above construction is the (*flat*) *torus*, which is obtained by gluing opposite faces in the unit cube $[0,1] \times [0,1] \times [0,1] \subset \mathbb{E}^3$. Specifically, the face $[0,1] \times [0,1] \times 0$ is identified to $[0,1] \times [0,1] \times 1$ by the translation map $(x,y,z) \to (x,y,z+1)$. The remaining pairs of faces $([0,1] \times 0 \times [0,1],\ [0,1] \times 1 \times [0,1])$ and $(0 \times [0,1] \times [0,1],\ 1 \times [0,1] \times [0,1])$ can be identified in an analogous way.

It is easy to check that the neighborhood of each point in $\mathbb{T}^3$ is a 3-ball of the Euclidean space. Thus $\mathbb{T}^3$ is indeed a 3-manifold, and its metric can be pushed from $\mathbb{E}^3$.

To compute a ray leaving a point $p \in \mathbb{T}^3$ in a direction v we use the ray $\mathcal{R}(t) = p + t \cdot v$ in $\mathbb{E}^3$. For each intersection between $\mathcal{R}$ and a face of the unit cube, we start again by replacing p by its correspondent point in the opposite face.

C. Manifolds as the quotient by discrete groups

We now take a more algebraic approach to build manifolds and special non-manifolds (*orbifolds*). A *simply connected space* is a connected topological space where each closed curve can be continuously deformed into a point. Examples 1, 2, and 3 are simply connected. The most famous problem related to such definition is Poincaré conjecture: the 3-sphere is the unique simple connected compact 3-manifold.

It is well know that for each connected manifold, there is a unique *simple connected universal covering* (Theorem 12.19 in [14]). Informally, a manifold $\widetilde{M}$ *covers* another manifold M if there is a map which "evenly covers" a neighborhood of each point in M. For example the torus is covered by the Euclidean space and the *projective space* (fundamental in computer graphics algorithms) is covered by the sphere. We now take the opposite direction: given a simple connected space and a group acting on it, we consider its quotient. By previous observation we only need to consider simply connected manifolds.

Here we follow the notations and definitions of [5], [14]. Let G be a group endowed with a topology and M be a manifold, if the *action* $p \to g(p)$, where $p \in M$ and $g \in G$, is a continuous map, then G is called a *continuous group*. A subgroup Γ of the continuous group G is called a *discrete group* if there is a neighborhood U of the G's identity such that $U \cap \Gamma$ is the identity element. For example the group of translation in $\mathbb{E}^3$ spanned by $(x,y,z) \to (x \pm 1, y, z)$, $(x,y,z) \to (x, y \pm 1, z)$, and $(x,y,z) \to (x,y,z \pm 1)$ is

a discrete group acting on $\mathbb{E}^3$. In particular, this group acts *freely*, since it leaves no fixed points.

Let M be a simply connected manifold and Γ be a discrete group acting on M, the *quotient* M/Γ is the set $\{\Gamma \cdot p |\ p \in M\}$ where $\Gamma \cdot p = \{g(p) |\ g \in \Gamma\}$ is the *orbit* of p. For example, the quotient of $\mathbb{E}^3$ by the group of translation cited above gives rise to the torus $\mathbb{T}^3$ presented in previous section. This implies in a new description of a ray $\mathcal{R}$ leaving a point $p \in \mathbb{T}^3$ in a direction v: $\mathcal{R}$ is described by considering the fractional part of the coordinates of the ray $\mathcal{R}(t) = p + t \cdot v$ in $\mathbb{E}^3$.

We are interested in the cases when M is a *geometry* space, so basically the spaces in Examples 1, 2, and 3. The quotient space M/Γ inherits the geometry of M. We say that M/Γ has the *geometric structure* modeled by M. For example, $\mathbb{T}^3$ has the geometric structure modeled by $\mathbb{E}^3$, in particular, for each $i, j, k \in \mathbb{Z}$ the unit cube $[i, i+1] \times [j, j+1] \times [k, k+1]$ in $\mathbb{E}^3$ is mapped isometrically to $\mathbb{T}^3$. Such cubes *tessellate* $\mathbb{E}^3$: this is what you actually see in an immersive view of $\mathbb{T}^3$.

The *fundamental domain* Δ plays an important role in the above construction. Δ is the region of M which contains exactly one point for each of these orbits $\{g(p) |\ g \in \Gamma\}$. The unit cube $[0, 1] \times [0, 1] \times [0, 1]$ is the fundamental domain of $\mathbb{T}^3$; note that it is not unique. Δ can be used to set the scene, and no deformation is required if we consider $M = \mathbb{E}^3$.

Let M be a manifold and Γ be a discrete group acting on it, when is M/Γ a manifold? *Quotient manifold theorem* (Theorem 9.16 in [14]) provides an answer. This informally states that M/Γ is a manifold when the (*Lie*) group Γ acts smoothly, freely, (and *properly*) on M. For example, if $M = \mathbb{E}^3$ then M/Γ is always the torus (Proposition 12.21 in [14]).

The *mirrored cube* $\mathcal{Q}^3$ is an example of a non-manifold with the geometric structure modeled by $\mathbb{E}^3$ through a special group of reflection Γ. Such group is generated by the reflections through the planes $x = \pm 1$, $y = \pm 1$, and $z = \pm 1$ in $\mathbb{E}^3$. The unit cube is the fundamental domain of $\mathcal{Q}^3$. Each time a ray $\mathcal{R}$ intersects a face of the fundamental domain of $\mathcal{Q}^3$ it is reflected, creating a polygonal curve in $\mathcal{Q}^3$: exactly what happen with the lights in a mirrored room. Such polygonal curve suspends to ray in $\mathbb{E}^3$, thus we see a tessellation of $\mathbb{E}^3$ by reflected unit cubes when immersed in $\mathcal{Q}^3$.

The mirrored cube $\mathcal{Q}^3$ is a particular example of an *orbifold*: space locally modelled by quotient of $\mathbb{E}^3$ by discrete groups. In Section VI we discuss future works using the spherical and hyperbolic geometry as covering spaces.

IV. GPU Ray Tracing of 3D Manifolds

In this section we present the method for immersive visualization of 3D manifolds and orbifolds using ray tracing on the GPU. For this purpose, we extended the work of Berger et al. [9] to the RTX GPUs.

A. Overview of the Method

The ray tracing algorithm is arguably the most natural method to produce visualizations of the intrinsic space of a 3D manifold/orbifold. Basically, its only necessary to adapt the traditional ray tracing of the Euclidean ambient space to take into account both the geometry and topology of the manifold/orbifold. The first aspect of this task is to simulate the ray path as it travels inside the space, starting from the point of observation until it intersects with a visible object. The second aspect amounts to shading that computes the illumination and evaluates the light scattered from the environment in the ray direction.

B. Algorithm in CPU

In order to understand these differences, let's study the basic ray tracing algorithm for polyhedral complexes that represent manifold/orbifold spaces – and compare it with the traditional ray tracing of Euclidean space.

As it can be verified in Algorithm 1, the rays are generated from the observer's point of view (lines 2 - 3) and intersected with visible objects (line 5) and if there is a hit (line 6), shading is done (line 7).

These three steps are present in all ray tracing algorithms including the traditional one for Euclidean space. In the case of ray tracing inside a manifold/orbifold we need extra steps to guide the path of a ray as it moves through the space. These correspond to lines 8, 9 and 4.

We assume that the whole computation has the *fundamental domain* as a base, which is modeled by a polyhedron Δ. Therefore, as the ray hits a face F_i in the boundary of the domain (line 8), we need to transport it by the action of the corresponding transformation of the discrete group (line 9).

For practical computational reasons we cannot continue the ray path indefinitely, thus a maximum level is set to stop the path (line 4).

Note that the most important and critical step is the group action (line 9), which is dependent of the geometry and topology of the manifold/orbifold. As such, it is specific for each different type of space.

Algorithm 1 Ray Tracing in manifolds/orbifolds

```
1:  for each pixel σ ∈ I do
2:      Let p := 0 and v be the direction associated to σ
3:      Generate a ray R from (p, v)
4:      for i ≤ level do
5:          Find the intersection i(R) with visible object O_0
6:          if i(R) ≠ ∅ then
7:              Paint pixel break
8:          else if The ray R intersects a face F_i of Δ then
9:              (⋆) Compute the new origin p' and ray R'.
10:         end if
11:     end for
12: end for
```

C. GPU RTX Pipeline

NVidia RTX is a hardware and software platform with support for real time ray tracing. The ray tracing code of an application using this architecture consists of CPU host code, GPU device code, the memory to transfer data to the Acceleration Structures for fast geometry culling when intersecting rays with scene objects.

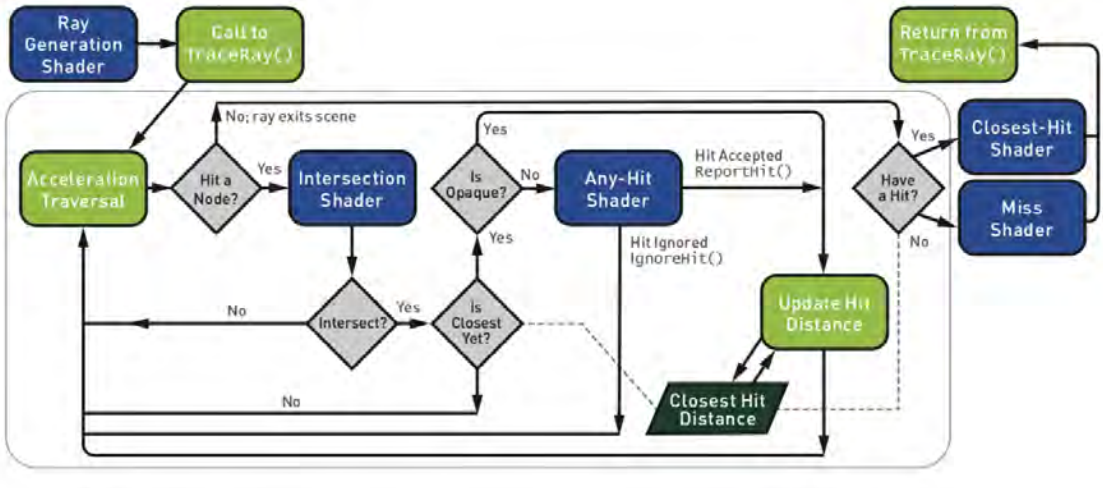

Fig. 1. Ray Tracing Pipeline (from DXR docs).

Specifically, the CPU host code manages the memory flow between devices, sets up, controls and spawn GPU shaders and defines the Acceleration Structures. Finally, the GPU role is to run instances of the ray tracing shaders in parallel. This is similar to the well-established rasterization rendering pipeline.

The ray tracing GPU device code runs in a similar pipeline scheme. The differences are the stages taken. The goal of the first stages is to generate the rays. Afterwards, a fixed intersection stage calculates the intersection of the rays with the scene geometry. Then, the intersection points are reported to the group of shading stages.

Each shader can be correlated with the tasks performed by the general CPU procedure described in Algorithm 1. The *Ray Generation Shader* is responsible for creating the rays, which are defined by their origins, directions and the custom user-defined data, called payloads (line 2). A call to TraceRay() launches a ray (line 3). The next stage is a fixed traversal of the Acceleration Structure. This traversal uses an *Intersection Shader* to calculate the intersections (line 5). All hits found pass by tests to verify if they are the closest hit.

After no additional hits are found, the *Closest-Hit Shader* is called for the closest intersection point (line 7). Figure 1 shows an in-depth scheme of the pipeline. More detailed information about RTX Ray Tracing can be seen in [15] and applications can be found in [16].

The above is the general ray tracing GPU pipeline. In the case of ray tracing inside a manifold/orbifold we have two classes of objects: i) the *scene objects* which are embedded in the space; and ii) the *boundary of the fundamental domain* that is represented by the polyhedron Δ. They are treated differently by the Closest-Hit Shader — while the scene objects are shaded in the regular way (line 7), the boundary of the fundamental domain transports the rays by the group action (line 9).

Another relevant point to notice, is that the fundamental domain acts as an additional acceleration structure for ray intersection with the objects in the scene.

D. Implementation

The application extends Ray-VR [2], an algorithm for stereo ray tracing built on top of Falcor [1], NVIDIA's scientific prototyping framework. It supports per-material effect customization, based on material ids. To simulate the torus and the mirrored cube immersive visualization we create the torus and perfect mirror shaders and material ids. The torus shader basically identifies when a ray intersects the boundary of the fundamental domain. The resulting point p and its normal vector N are used to update p by $p - d \cdot N$ in the opposite face, where d is the fundamental domain diameter. The ray tracing continues from this new point. The perfect mirror shader works in the obvious way by considering the cube faces as perfect mirrors.

We use the software Blender to create the fundamental domains, including their boundaries and scene objects. The hardware setup consists of a computer with a NVIDIA GeForce 2080 Ti for RTX Ray Tracing support and a HTC Vive for VR visualization.

V. Examples and Results

In this section we present some expressive output images from our implementation of the torus and the mirrored cube spaces, examples of manifold and orbifold, respectively.

A. Manifolds

Figure 2 presents the visualization of the 3-dimensional torus $\mathbb{T}^3$ using the torus shader described in Subsection IV-D. There is only one monkey's head, the *Suzanne* classical Blender mesh. We attach Suzanne to the camera. The many Suzanne's copies are due to the fact that rays in $\mathbb{T}^3$ can be closed curves. For a more algebraic explanation, this image describes the action of the group of translation in the Euclidean space which coves $\mathbb{T}^3$, explaining thus the copies pattern.

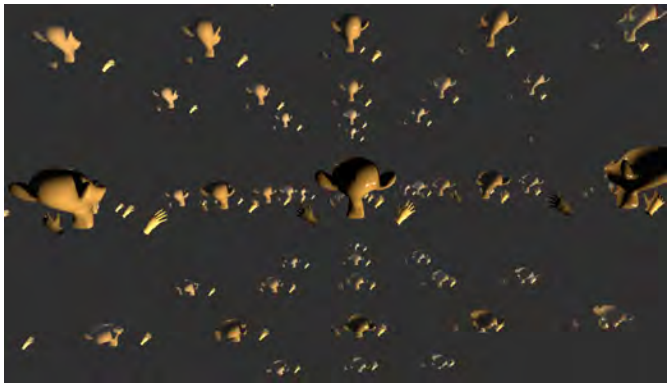

Fig. 2. 3D Torus

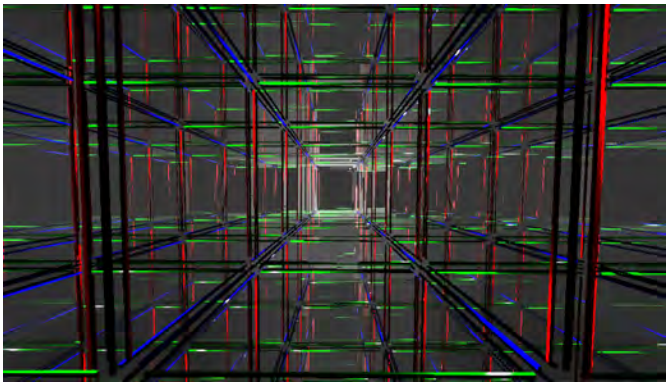

Fig. 4. Space Structure

Figure 3 illustrates an immersive visualization of the mirrored cube using the perfect mirror shader (Subsection IV-D). Again, there is a single Suzanne in the scene attached to the camera. The image is the view of the group of reflection, generated by the cube's faces, acting on the Euclidean space.

Fig. 3. Mirror Cube

B. Space Perception

To produce a better understanding of the torus and mirrored cube space structure, we add the edges of the fundamental domain to the scene, see Figure 4. The result gives a perception of a tessellation of the Euclidean space.

In this example, the complete cell structure of the covering space is readily apparent since we explicitly marked the boundary of the fundamental domain.

More subtle perception arise if only some static objects are placed in key landmarks of the domain. Moreover, adding a dynamic behavior may give a transient or pulsating character to the space (i.e., with random or periodic motion, respectively).

In addition, when the viewer is placed inside an opaque cell with a few openings (e.g., a cube with doors and windows), the perception of an infinite space changes to that of a maze.

One important ingredient in the understanding of the space structure is the scale, which is related to the fundamental domain volume. In Figure 4, for example, we are able to see many copies of the fundamental domain, which produce, again, the view of its covering space. However if we consider a fundamental domain sufficiently larger, the user will be able to visualize mostly the immediate surroundings of the scene restricted to the fundamental domain. This leads us to a philosophical question: *could we be living inside a 3D torus?* or *what is the shape of the Universe?*

C. Interaction

To give the user a better perception of the torus and the mirrored room, we attach, besides Suzanne to the camera, models of the left and right hands to the left and right controls of the HTC Vive (see Figure 3). Thus interacting in the fundamental domain provides a better sense of being immersed in the quotient spaces.

Future works include the motion capture of the user whole body skeleton, using techniques reminiscent from computer vision and artificial intelligence [17] (see Figure 5).

Fig. 5. Pose Detection and Motion Capture

VI. Extensions and Future Work

This work opens many questions related to using virtual reality to visualize abstract spaces. Basically, if in such spaces we are able to compute rays (geodesics) and their intersection with embedded objects (submanifolds, probably), then there is some hope in visualizing such space.

A. Other Spaces and Manifolds

As commented in Subsection III-B, the torus $\mathbb{T}^3$ is the unique manifold having the geometric structure modeled by $\mathbb{E}^3$. However if we consider the spherical/hyperbolic spaces, many manifolds can be constructed. In dimension two, for example, the *Uniformization theorem* states that the hyperbolic plane covers every 2-manifold with negative Euler characteristic. The analogous result for dimension three is the *Thurston–Pereman geometrization theorem* which states that every manifold has a certain geometric structure that can be associated to it [3]. However, in this case, it is not possible to associate a single geometry to the whole manifold, the geometrization theorem states that we can decompose the manifolds in parts, each of these with geometric structure modeled by one of the eight (Thurston) geometries. These include Examples 1, 2, and 3, we skip the presentation of the others five models.

In future works we pretend to explore the visualization of 3-manifolds having structures modelled by other geometries. Famous examples of such manifolds include the *Poincaré dodecahedron space* and *Seifert-Weber dodecahedral space* [3]. The first one, discovered by Poincaré, is obtained by gluing the opposite faces of a dodecahedron embedded in the spherical space $\mathbb{S}^3$. Considering the dodecahedron embedded in the hyperbolic space $\mathbb{H}^3$, a special clockwise identification of its opposite faces gives rise to Seifert-Weber dodecahedral space. Both manifolds have their geometric structures modelled by $\mathbb{S}^3$ and $\mathbb{H}^3$, respectively.

For an example of a non-manifold with geometric structure modeled by the hyperbolic space, consider the dodecahedron embedded in $\mathbb{H}^3$. Let Γ be the group of reflections generated by the dodecahedral faces. The quotient $\mathbb{H}^3/\Gamma$ is the *mirrored dodecahedral space*. In [9], the authors provided an immersive view of such space, the images are inspiring.

To visualize the spaces described above, basically, we have to compute a ray in $\mathbb{H}^3$ or $\mathbb{S}^3$, which can be presented by arcs and parabola. Thus we need a hit shader that computes the intersection between bounding boxes and such curves.

B. Space and Time Effects

Fog is a technique used in rendering to enhance the space perception by letting the object shading to be dependent of its distance from the camera, assuming a participatory medium. Figure 6 presents the effects caused when applying fog in the space: its left side did not received fog, while the right side received.

Adding a delay during the ray tracing would produce a significant contribution to the space perception.

In future works we pretend to let the light rays travel among cells, increasing their contribution of the scene shading. For now, it is affecting the algorithm performance.

For a more artistic application, the scene shading could depend on the cell (thinking that the rays travel in the space covering). Each cells in the torus covering space has a code $i, j, k \in \mathbb{Z}$, as described in Subsection III-C. Thus adding some on-off probabilistic rule depending on the integers i, j, k could provide a special effect to be exploited by artistic control.

C. Applications

For a better user experience in Non-Euclidean spaces, well-designed *tours* and *games* are desired within these spaces.

For a tour, the virtual user body could be attached to an object, a airplane for example, which is controlled by the HTC Vive controls. This would allow the user to travels among cells of the covering space.

Extending the tour, a game inside these abstract spaces would produce "surreal" experience. Since computing rays is our speciality, a "shooter" game would be a first candidate. Adding some on-off probabilistic rule depending on the cell could give increasing challenge to the player, based on his/her level in the game. Such application is an extension of the two dimensional case presented in [11].

There is also the possibility of many users interacting in a same space. For that, we intend to use the visual motion capture system [17] mentioned in the previous section and the interaction framework of Velho et al. [18].

VII. Conclusion

In this paper, based on ray tracing algorithms, we introduced a real–time immersive visualization of some three dimensional manifolds and orbifolds. The algorithm was based on Falcor, NVIDIA's scientific prototyping framework, which uses the power of the new generation of RTX GPUs.

For a theoretical point of view, our framework could be used to investigate abstract phenomena in geometry and topology of low dimension. However our work goes beyond this, it establishes new possibilities for the use of Non-Euclidean space in games, art, and in the dissemination of those exotic abstract spaces.

References

[1] N. Benty, K.-H. Yao, T. Foley, M. Oakes, C. Lavelle, and C. Wyman, "The Falcor rendering framework," 05 2018, https://github.com/NVIDIAGameWorks/Falcor.

[2] Anonymous, "Ray tracing virtual reality in falcor : Ray-vr," —, Technical Report TR-xx-2019, 2019.

[3] W. Thurston, *The geometry and topology of three-manifolds*. Princeton University, 1979. [Online]. Available: http://books.google.com.br/books?id=FDcZAQAAIAAJ

[4] N. Amenta, S. Levy, T. Munzner, and M. Phillips, "Geomview: a system for geometric visualization," in *Proceedings of the eleventh annual symposium on Computational geometry*, ser. SCG '95. New York, NY, USA: ACM, 1995, pp. 412–413. [Online]. Available: http://doi.acm.org/10.1145/220279.220327

[5] C. Gunn, "Discrete groups and visualization of three-dimensional manifolds," in *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ser. SIGGRAPH '93. New York, NY, USA: ACM, 1993, pp. 255–262. [Online]. Available: http://doi.acm.org/10.1145/166117.166150

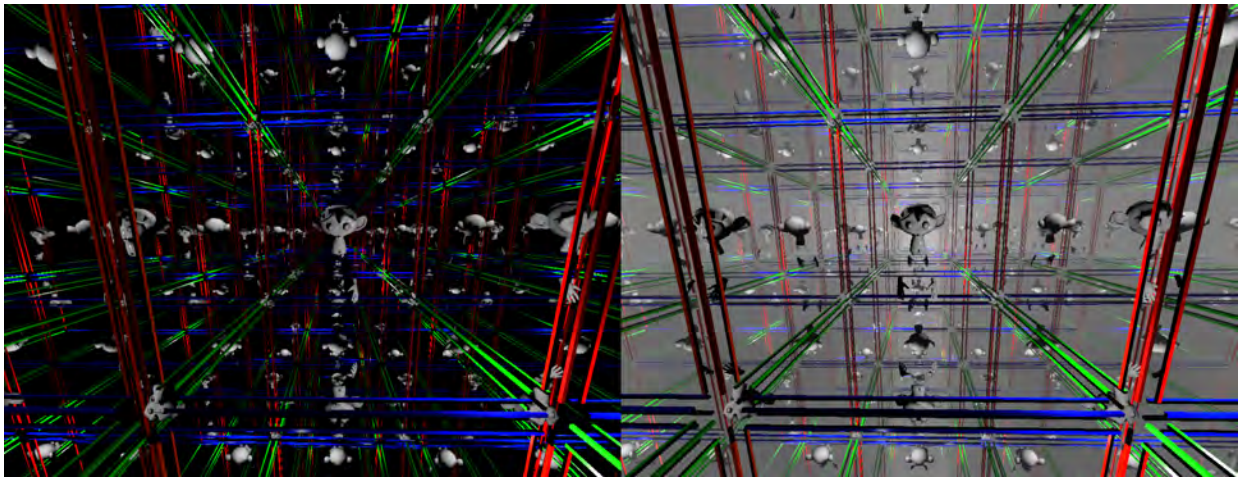

Fig. 6. Fog

[6] R. Hudson, C. Gunn, G. K. Francis, D. J. Sandin, and T. A. DeFanti, "Mathenautics: using vr to visit 3-d manifolds," in *Proceedings of the 1995 symposium on Interactive 3D graphics*, ser. I3D '95. New York, NY, USA: ACM, 1995, pp. 167–170. [Online]. Available: http://doi.acm.org/10.1145/199404.199433

[7] G. K. Francis, C. M. A. Goudeseune, H. J. Kaczmarski, B. J. Schaeffer, and J. M. Sullivan, "Alice on the eightfold way: Exploring curved spaces in an enclosed virtual reality theatre," in *Visualization and Mathematics III*, 2003, pp. 305–315.

[8] S. Weismann, C. Gunn, P. Brinkmann, T. Hoffmann, and U. Pinkall, "jreality: a java library for real-time interactive 3d graphics and audio." in *ACM Multimedia'09*, 2009, pp. 927–928.

[9] P. Berger, A. Laier, and L. Velho, "An image-space algorithm for immersive views in 3-manifolds and orbifolds," *Visual Computer*, 2014.

[10] C. Gunn, "Advances in Metric-neutral Visualization," in *GraVisMa 2010*, V. Skala and E. Hitzer, Eds., Eurographics. http://gravisma.zcu.cz/GraVisMa-2010/GraVisMa-2010-proceedings.pdf, 2010, pp. 17–26.

[11] F. Guimaraes, V. Mello, and L. Velho, "Geometry independent game encapsulation for non-euclidean geometries," in *Proceedings of SIBGRAPI Workshop of Works in Progress*, 2015.

[12] J. M. Lee, "Smooth manifolds," in *Introduction to Smooth Manifolds*. Springer, 2013, pp. 1–31.

[13] A. T. c. Fomenko and S. V. Matveev, *Algorithmic and computer methods for three-manifolds*. Springer Science & Business Media, 2013, vol. 425.

[14] J. Lee, *Introduction to topological manifolds*. Springer Science & Business Media, 2010, vol. 202.

[15] C. Wyman, S. Hargreaves, P. Shirley, and C. Barr-Brisebois, "Introduction to DirectX Raytracing," in *ACM SIGGRAPH 2018 Courses*, Aug. 2018. [Online]. Available: http://intro-to-dxr.cwyman.org/

[16] E. Haines and T. Akenine-Mller, Eds., *Ray Tracing Gems*. Apress, 2019.

[17] Anonymous, "Tensorpose: Real-time pose estimation using tensorflow for interactive applications," —, Technical Report TR-yy-2019, 2019.

[18] L. Velho, L. Carvalho, and D. Lucio, "Vr tour: Guided participatory meta-narrative for virtual reality exploration," VISGRAF Lab - IMPA, Technical Report TR-06-2018, 2018.