Gaming in Elliptic Geometry

László Szirmay-Kalos and Milán Magdics

Budapest University of Technology and Economics, Dept. of Control Engineering and Information Technology (e-mail: szirmay@iit.bme.hu)

Abstract

An interesting way to explore curved spaces is to play games governed by the rules of non-Euclidean geometries. However, modeling tools and game engines are developed with Euclidean geometry in mind. This paper addresses the problem of porting a game from Euclidean to elliptic geometry. We consider primarily the geometric calculations and the transformation pipeline.

1. Introduction

Euclidean, elliptic and hyperbolic geometries differ only in the *parallel axiom*, i.e., for a given line and point not on the line, they postulate the existence of exactly one, none, and more than one non-intersecting line passing through the given point. Visualizing non-Euclidean spaces is important in mathematics, cartography and art [LR96, GMV15, OCH19]. Image space rendering reinterprets the definition of a ray according to the geodesics of the geometry [Gro95, BLV15, VSN20, WBE*06]. Object space rendering algorithms can be adapted realizing that ignoring metric properties like distance or angle, projective geometry is a common ground for all geometries [Wee02, Gun10]. However, animation of objects and the camera definition involve geometric calculations like the determination of distance, angle and direction, finding orthogonal directions, and setting up transformation matrices [PG92, HHMS17, MN19].

The objective of this paper is to present a simple method of converting games developed for Euclidean geometry to elliptic geometry. We consider the modification of the matrices in the full transformation pipeline, the conversion of the objects from Euclidean to elliptic space, and geometric calculations.

2. The embedding space

To examine Euclidean and elliptic geometries analytically, we can take an outsider's view, and look at them from a space of one more dimensions, i.e., we consider the 3D geometries as subsets of a 4D *embedding* space. The embedding space is associated with four orthogonal unit basis vectors **i**, **j**, **k**, **l**, and its elements are characterized by four-element *row vectors* $\mathbf{v} = (x, y, z, w)$. The point **g** of coordinates $(0,0,0,1)$ is included in both geometries, and is called the *geometry origin* to distinguish it from the embedding space *origin* of coordinates $(0,0,0,0)$. The 4D embedding space is endowed with *dot product*:

$$\langle \mathbf{v}_1, \mathbf{v}_2 \rangle = x_1 x_2 + y_1 y_2 + z_1 z_2 + w_1 w_2.$$

2.1. Euclidean geometry

In the embedding space, the points $\mathbf{p} = (p_x, p_y, p_z, p_w)$ of the *Euclidean 3D space* are identified by equation $p_w = 1$. As vectors are directions between two points, vectors **v** of the Euclidean space have $v_w = 0$. A line defined by points **p** and **q** is the intersection of the 3D Euclidean space and the 2D plane defined by points **p**, **q** and the origin of the embedding space (Figure 1). The 4D space can also embed the *Projective 3D geometry* if we consider lines crossing the origin to be the points of the projective space.

2.2. Elliptic geometry

Points **p** belong to the *elliptic 3D geometry* if they are on the unit 3D hyper-sphere defined by equation $\langle \mathbf{p}, \mathbf{p} \rangle = 1$. Points of the elliptic geometry are *diameters* of the unit hyper-sphere, thus antipodal points (p_x, p_y, p_z, p_w) and $(-p_x, -p_y, -p_z, -p_w)$ are the same. Treating points as diameters preserves the validity of the Euclidean axiom stating that *"two distinct points unambiguously define a line"* also in the spherical structure. Elliptic geometry is in fact projective geometry with the metric of the sphere. To guarantee that antipodal points provide the same results in calculations, we take the absolute value of the dot product.

Vectors **v** are directions that do not point out of the geometry, thus they must be in the 3D tangent hyperplane called *tangent space* of the hyper-sphere at their start. Position vectors of points on an arbitrary dimensional sphere centered in the origin are orthogonal to the tangent space, thus vectors should satisfy $\langle \mathbf{p}, \mathbf{v} \rangle = 0$.

A *line* defined by points **p** and **q** is again the intersection of the 3D hyper-sphere and the 2D plane of the embedding space defined by points **p**, **q** and its origin similarly to Euclidean geometry. It means that lines in Euclidean and elliptic geometries are projectively equivalent assuming the origin of the embedding space to be the center of projection.

The *distance d* between points **p** and **q** is the angle between their directions from the origin, i.e. $d = \cos^{-1}(|\langle \mathbf{p}, \mathbf{q} \rangle|)$.

Figure 1 shows the 2D projective, Euclidean and elliptic geometries in the 3D embedding space. Note that this figure is for analogy only since we investigate 3D geometries in the 4D embedding space, which would be hard to visualize.

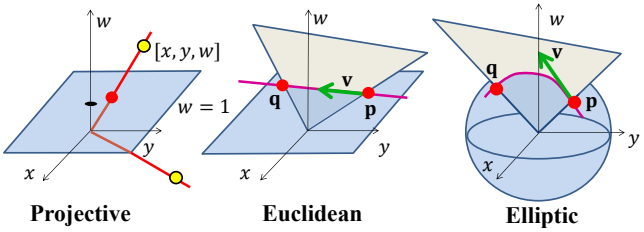

Figure 1: *2D Euclidean points are in the $p_w = 1$ plane while elliptic points are on the $\langle \mathbf{p}, \mathbf{p} \rangle = 1$ sphere. Vectors are in the tangent plane of their starting point. A line defined by points* **p** *and* **q** *is on the plane defined by* **p**, **q** *and the origin. 2D projective geometry considers the lines crossing the origin to be points of the geometry to reduce the dimension of the embedding space.*

3. Finding a vector that is orthogonal to two other vectors

The calculation of vector **v** that is orthogonal to two other vectors **a** and **b** is needed in many tasks, including the construction of the view matrix or the Frenet frame. In Euclidean geometry, the cross product solves this problem, so we are looking for its appropriate generalization in elliptic geometry by taking into account that the result is a vector that should also be in the tangent space of point **p** where this calculation is made, i.e., orthogonal to position vector **p**. The proposed operation is called the *three-operand cross product*:

$$\mathbf{v} = \otimes(\mathbf{p}, \mathbf{a}, \mathbf{b}) = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} & \mathbf{l} \\ p_x & p_y & p_z & p_w \\ a_x & a_y & a_z & a_w \\ b_x & b_y & b_z & b_w \end{vmatrix}. \tag{1}$$

To prove that vector **v** is indeed perpendicular to the three operands, let us consider the scalar product of **v** and an arbitrary vector **u**:

$$\langle \mathbf{u}, \mathbf{v} \rangle = \begin{vmatrix} u_x & u_y & u_z & u_w \\ p_x & p_y & p_z & p_w \\ a_x & a_y & a_z & a_w \\ b_x & b_y & b_z & b_w \end{vmatrix}.$$

If **u** were in the subspace of **p**, **a**, **b**, then according to the properties of determinants, the determinant would be zero, so the dot product of the result of Eq. 1 and **u** is zero, indicating that **v** is in the tangent space at **p** and orthogonal to **a** and **b**.

4. Transformations and isometries

Our goal is to express transformations as 4×4 matrix multiplications, which should map the set of points of the elliptic geometry onto the same set, and preserve antipodal equivalence. 4×4 matrices of the $O(4)$ isometry group, where rows are orthogonal unit vectors, meet this criterion.

4.1. Translation

In Euclidean geometry translating an object means adding the translation vector to the position vector of the vertices. However, in elliptic geometry, the allowed vectors depend on the point, thus we have to find an alternative in the $O(4)$ group. To identify the appropriate class of isometries that can be called translations in elliptic geometry, we consider the intuitive properties. The translation

- is an isometry preserving the dot product and the orientation,
- is defined by point $\mathbf{q} = (q_x, q_y, q_z, q_w)$ to which the geometry origin of coordinates $\mathbf{g} = (0,0,0,1)$ is translated, and
- should keep the original and modified direction parallel in the embedding space as much as possible.

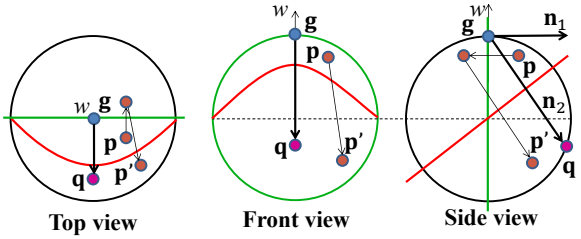

Figure 2: *Translation is the composition of two reflections on hyperplanes crossing the origin and of normals $\mathbf{n}_1 = (q_x, q_y, q_z, 0)$ and $\mathbf{n}_2 = \mathbf{q} - \mathbf{g}$, respectively.*

Concerning the requirement of isometry and orientation preservation, we are left with 4D rotations or equivalently even number of reflections. In order to distinguish elliptical translation from elliptical rotation, we can say that elliptical translations modify the geometry origin and keep the directions as parallel as possible while elliptical rotations preserve the geometry origin. Both translation and rotation can be built of even number of reflections. In Euclidean geometry, the composition of two reflections is a translation if the two planes are parallel. In our case, the transformation should preserve the hyper-sphere, therefore the planes should contain the origin, thus they cannot be parallel, but we can aim at the closest possible case. It means that we use planes that are orthogonal to the geodesic between the geometry origin and target point **q**. Reflection of point **p** on a hyperplane of normal vector **n** in the 4D embedding space can be obtained as

$$\mathbf{p}' = \mathbf{p} - 2\frac{\langle \mathbf{p}, \mathbf{n} \rangle}{\langle \mathbf{n}, \mathbf{n} \rangle}\mathbf{n} \tag{2}$$

which is a linear operation for **p** and can thus be expressed as a matrix multiplication. We use two reflections transforming the geometry origin **g** to **q** (Figure 2). The first hyperplane goes through the geometry origin and has a normal vector that points into the direction of **q** on the sphere, i.e. $\mathbf{n}_1 = (q_x, q_y, q_z, 0)$. The second hyperplane is halfway between the geometry origin and point **q** with normal vector $\mathbf{n}_2 = \mathbf{q} - \mathbf{g}$. Substituting these into Eq. 2, we obtain:

$$\mathbf{T}(\mathbf{q}) = \begin{bmatrix} 1 - \frac{q_x q_x}{1+q_w} & -\frac{q_x q_y}{1+q_w} & -\frac{q_x q_z}{1+q_w} & -q_x \\ -\frac{q_y q_x}{1+q_w} & 1 - \frac{q_y q_y}{1+q_w} & -\frac{q_y q_z}{1+q_w} & -q_y \\ -\frac{q_z q_x}{1+q_w} & -\frac{q_z q_y}{1+q_w} & 1 - \frac{q_z q_z}{1+q_w} & -q_z \\ q_x & q_y & q_z & q_w \end{bmatrix}.$$

The last row of the matrix is the target point $\mathbf{q}$, thus this matrix indeed translates the geometry origin to this point.

4.2. Rotation

Rotation is also an isometry, but unlike translation, it keeps the geometry origin at $(0,0,0,1)$. From this, the fourth row of the transformation matrix should also be $(0,0,0,1)$. In case of isometries the row vectors of the matrix are orthogonal, thus the first three row vectors must be orthogonal to $(0,0,0,1)$, which means that their fourth coordinates are zero. So, the structure of the transformation matrix in elliptic geometry is identical to that of the Euclidean geometry.

4.3. The view matrix

The camera is defined by eye position $\mathbf{e}$ and three orthogonal unit vectors in the tangent space of the eye, right direction $\mathbf{i}'$, up direction $\mathbf{j}'$, and negative view direction $\mathbf{k}'$. The orthogonality can be enforced with the *three-operand cross product* of Section 3. The view matrix transforms the coordinates of this basis to the basis at the geometry origin where the axes are $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$, $\mathbf{l}$. The view matrix can be expressed as

$$\mathbf{V} = \begin{bmatrix} i'_x & j'_x & k'_x & e_x \\ i'_y & j'_y & k'_y & e_y \\ i'_z & j'_z & k'_z & e_z \\ i'_w & j'_w & k'_w & e_w \end{bmatrix} \tag{3}$$

To prove that this matrix meets the requirements of the view matrix, we look at the transformation of the eye position $\mathbf{e}$ and the basis vectors. The eye position is transformed as

$$\mathbf{e} \cdot \mathbf{V} = (\langle \mathbf{e}, \mathbf{i}' \rangle, \langle \mathbf{e}, \mathbf{j}' \rangle, \langle \mathbf{e}, \mathbf{k}' \rangle, \langle \mathbf{e}, \mathbf{e} \rangle) = (0,0,0,1).$$

since $\mathbf{i}'$, $\mathbf{j}'$, $\mathbf{k}'$ are in the tangent space of eye position $\mathbf{e}$, and the eye position is in the elliptic space identified by $\langle \mathbf{e}, \mathbf{e} \rangle = 1$.

The transformation of the right direction $\mathbf{i}'$ is

$$\mathbf{i}' \cdot \mathbf{V} = (\langle \mathbf{i}', \mathbf{i}' \rangle, \langle \mathbf{i}', \mathbf{j}' \rangle, \langle \mathbf{i}', \mathbf{k}' \rangle, \langle \mathbf{i}', \mathbf{e} \rangle) = (1,0,0,0) = \mathbf{i}.$$

The transformation of $\mathbf{j}'$ and $\mathbf{k}'$ is similar.

4.4. The perspective transformation matrix

After the view transformation, the camera is at the geometry origin, looks at the $-z$ direction, its right direction is axis x and its up direction is axis y. The visible frustum is defined by field of view angles δ_x and δ_y as well as minimal $d_{\min}$ and maximal $d_{\max}$ distances on the *optical axis*, i.e. along the geodesic leaving the geometry origin in the $-z$ direction. Using OpenGL the GPU assumes that the vertex shader outputs the point in homogeneous coordinates and the viewing rays are parallel with axis z, and considers a point inside the frustum if inequalities $-w \le x, y, z \le w$ are satisfied. Thus, the perspective transformation should map the selected frustum to the domain defined by the clipping inequalities.

With a linear transformation of the 4D embedding space, the 3D hyper-sphere of elliptic geometry is transformed to a 3D hyper-ellipsoid (Figure 3). Let us consider three special points on the circular optical axis: The eye position is the geometry origin $\mathbf{g} = (0,0,0,1)$. The entry point in the view frustum is $(0,0,-\sin(d_{\min}),\cos(d_{\min}))$, and the exit point is $(0,0,-\sin(d_{\max}),\cos(d_{\max}))$. The eye position should be mapped to the ideal point $(0,0,\lambda,0)$ of axis z, the entry point to the front clipping plane defined by $-w' = z'$ and the exit point to the back clipping plane of equation $w' = z'$. From these requirements, we can obtain the following perspective transformation matrix

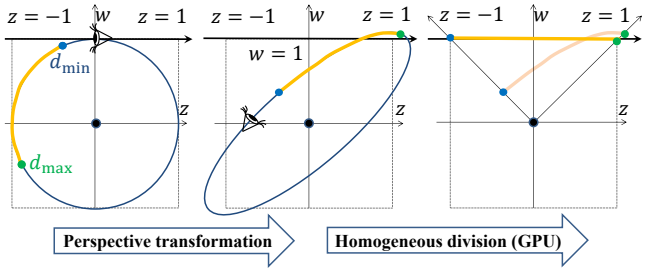

Figure 3: *Perspective transformation maps the hyper-sphere to hyper-ellipsoid preparing the object for GPU clipping and projection, which interprets embedding coordinates as homogeneous coordinates.*

$$\mathbf{P} = \begin{bmatrix} \frac{1}{\tan(\delta_x/2)} & 0 & 0 & 0 \\ 0 & \frac{1}{\tan(\delta_y/2)} & 0 & 0 \\ 0 & 0 & -\frac{\sin(d_{\min}+d_{\max})}{\sin(d_{\max}-d_{\min})} & -1 \\ 0 & 0 & -\frac{2\sin(d_{\min})\sin(d_{\max})}{\sin(d_{\max}-d_{\min})} & 0 \end{bmatrix}.$$

5. Porting objects from Euclidean to elliptic geometry

When the virtual world is created, we usually use modeling tools following the rules of Euclidean geometry, and outputting points in Cartesian coordinates. The models and points need to be "transported" to the elliptic space preserving all properties that are valid in both geometries. We assume that our objects are also triangle meshes in elliptic geometry, where a triangle is defined by its vertices, and edges are straight line segments. So, we need to consider only the porting of points representing vertices and the vectors, e.g. normals, associated with them. As lines only approximately preserved, the tessellation should be refined enough.

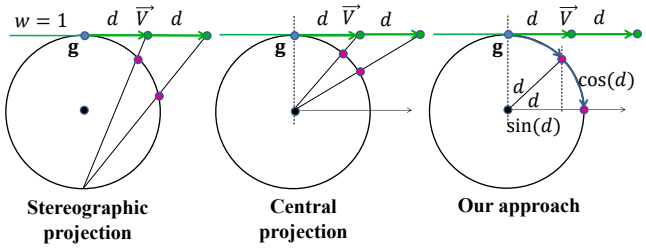

Figure 4: *Transporting objects from Euclidean to elliptic space. Stereographics projection preserve angles and circles, central projection lines, but both of them strongly distort distances.*

As the spherical geometry has non-zero curvature, any correspondence with Euclidean geometry necessarily introduces distor-

tions, such as change of distance, angle, or type of geometric primitives. There are many possibilities to project the Euclidean space onto a sphere or a half-sphere (Figure 4). Instead of mapping the infinite Euclidean space to the finite elliptic space, we need to transform a finite part of the Euclidean space to the hyper-sphere, but only with distortions really required by the curved space. As in games, objects are typically defined in modeling space, i.e. close to the origin, and transformed to the actual position by translations, we prefer a mapping where the distortion diminishes close to the geometry origin.

A mapping meeting this requirement is based on the recognition that it is worth preserving the distance and the direction of the point from the geometry origin. If a point has Cartesian coordinates $\vec{P} = [X,Y,Z]$ in Euclidean geometry, it means that the point is at distance $d = \sqrt{X^2+Y^2+Z^2}$ from the geometry origin and its direction is defined by unit vector $\vec{P}/d$. The natural pair of this Euclidean point in elliptic space is the point

$$\mathbf{p} = \mathcal{P}(\vec{P}) = \left(\vec{P}/d \sin(d), \cos(d)\right)$$

that is also in direction $\vec{P}/d$ and at the same distance d from the geometry origin. Note that the tangent space of elliptic and Euclidean geometries are identical at the geometry origin, thus this direction can be ported from one to the other without any modification.

Let us consider vector $(\vec{V},0)$ in Euclidean space starting at point $(\vec{P},1)$. If point $(\vec{P},1)$ were at the geometry origin, then the Euclidean and the elliptic space vectors would be the same. If the point is moved to $\mathbf{p}$, the vector should follow it with minimum change, which is provided by the developed matrix of translation. This means that vectors can be transported from Euclidean to elliptic space by applying the translation on them:

$$\mathbf{v} = (\vec{V},0) \cdot \mathbf{T}(\mathcal{P}(\vec{P})). \tag{4}$$

6. Game adaptation and results

The proposed method has also been integrated into the Unity3D engine. Object definition can be adapted according to Section 5, which needs the modification of an existing game engine where it uploads triangle meshes to the GPU. The transformation matrices are changed according to the results of Section 4, which can happen in the engine where these matrices are passed as uniform variables to the GPU. Considering diameters as "points" in elliptic geometry means that an object is visible in the location mirrored at the origin as well. This effect is produced by rendering every object twice, once with the original coordinates and once with negated ones.

Figure 5 shows snapshots from games comparing the feeling of the Euclidean and elliptic geometries.

7. Conclusion

This paper proposed an approach to prepare games in elliptic geometry. We investigated the adaptation of the geometric calculations, object definitions, transformation matrices and the physical simulation.

Acknowledgements

This project has been supported by OTKA K-124124.

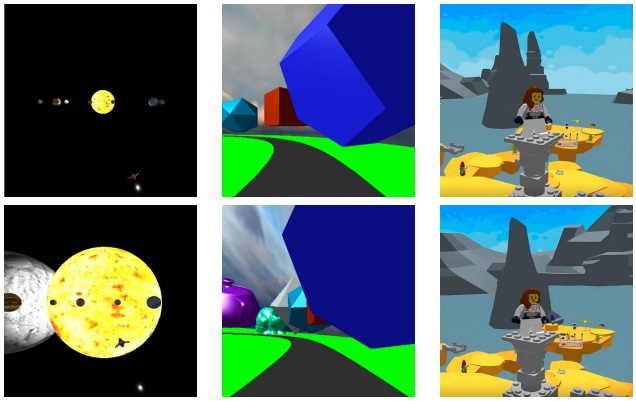

Figure 5: *Games in Euclidean (top) and in elliptic (bottom) spaces. Note that in elliptic space objects at distance close to* π *have similar perceived size as objects being at distance close to zero.*

References

[BLV15] BERGER P., LAIER A., VELHO L.: An image-space algorithm for immersive views in 3-manifolds and orbifolds. *The Visual Computer 31* (2015), 93–104.

[GMV15] GUIMARAES F., MELLO V., VELHO. L.: Geometry independent game encapsulation for non-euclidean geometries. In *Proceedings of SIBGRAPI* (2015).

[Gro95] GROELLER E.: Nonlinear ray tracing: visualizing strange worlds. *The Visual Computer 11*, 5 (1995), 263–274.

[Gun10] GUNN C.: Advances in metric-neutral visualization. In *GraVisMa* (2010), pp. 17–26.

[HHMS17] HART V., HAWKSLEY A., MATSUMOTO E. A., SEGERMAN H.: Non-euclidean virtual reality I: explorations of H^3, 2017. http://arxiv.org/abs/1702.04004.

[LR96] LAMPING J., RAO R.: The hyperbolic browser: A focus+context technique for visualizing large hierarchies. *Journal of Visual Languages & Computing 7*, 1 (1996), 33 – 55.

[MN19] MCCALEB REACH A., NORTH C.: Smooth, efficient, and interruptible zooming and panning. *IEEE Transactions on Visualization and Computer Graphics 25*, 2 (2019), 1421–1434.

[OCH19] OSUDIN D., CHILD C., HE Y.-H.: Rendering non-euclidean space in real-time using spherical and hyperbolic trigonometry. *Computational Science – ICCS 2019* (2019), 543–550.

[PG92] PHILLIPS M., GUNN C.: Visualizing hyperbolic space: Unusual uses of 4x4 matrices. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics* (1992), I3D '92, pp. 209–214.

[VSN20] VELHO L., SILVA V. D., NOVELLO T.: Immersive visualization of the classical non-euclidean spaces using real-time ray tracing in VR. In *Proceedings of Graphics Interface* (2020), pp. 423–430.

[WBE*06] WEISKOPF D., BORCHERS M., ERTL T., FALK M., FECHTIG O., FRANK R., GRAVE F., KING A., KRAUS U., MULLER T., NOLLERT H., MENDEZ I. R., RUDER H., SCHAFHITZEL T., SCHAR S., ZAHN C., ZATLOUKAL M.: Explanatory and illustrative visualization of special and general relativity. *IEEE Transactions on Visualization and Computer Graphics 12*, 4 (2006), 522–534.

[Wee02] WEEKS J.: Real-time rendering in curved spaces. *IEEE Computer Graphics and Applications 22*, 6 (2002), 90–99.

[Wei01] WEISKOPF D.: *Visualization of Four-Dimensional Spacetimes*. PhD thesis, University of Tübingen, Germany, 2001.