# Advanced Computer Graphics –Car Solo Ride Mario Stadium

**Chen Yen Lin**[1]

*D12458002 Institute of Medical Device and Imaging, National Taiwan University, College of Medicine.*

**Abstract:** In this work, a taxi car is driven in counterclockwise direction along the race track in the Mario stadium. The car needs to be driven in the left or right lane in the reasonable animation. The matrix transformation is applied based on given track data and normal direction. Theoretical method and modeling result would be demonstrated in this paper.

***Key Word**—Three.JS, HTML, JavaScript, Visual Studio Code.*

## 1. Introduction

Before the solo race, a obj file loader, .GLTF and xyz data loader can be found on the three.js web site on the fundamental section. As a Taxi car is driven along the track in the Mario stadium. The position of car is changed according the xyz track data. After complete one loop, the loop will restart again. In order to get a correct orientation along the track, the taxi would rotate around x-axis, y-axis and z-axis at a certain amount based on the traveling direction and normal direction. There are several methods to complete a track with correct and orientation, some function such as quaternion.setFromAxisAngle, lookAt, and matrix4 can be combined to set up correct orientation along the track. In the very beginning, the software MeshLab is being used to visualize the xyz track data and normal vectors. In addition, the car orientation is being rotated around z-axis at 180 degrees to match the y coordinate of the Mario stadium before importing the program. The orientation of the car is shown in Fig.1. Therefore, the texture mapping image is rotated too. The obj file and mtl file has to be saved in the different file name.

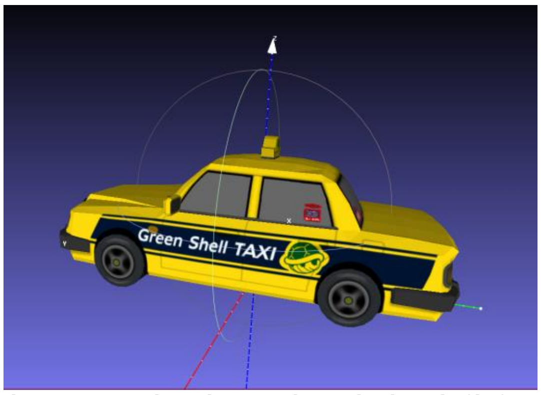
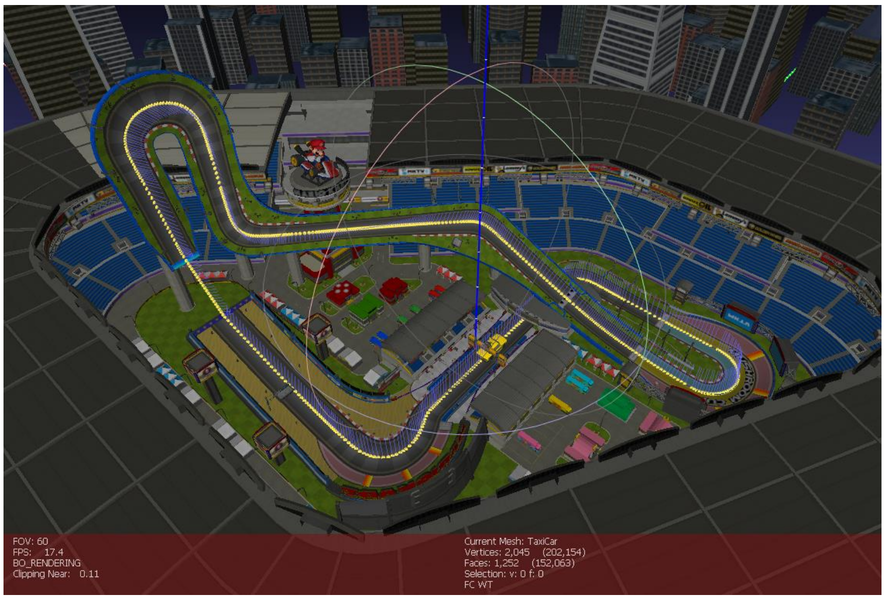


**Fig. 1** The taxis car mode in shown in the MeshLab in obj file format.

## 2. Theory and Methods

After importing the track data, Taxi car and Mario stadium, we need to move the car around the race track. The traveling direction of the car can be determined by the 3D points in the track data. The tangent of the curve is also can be the representation of the car moving direction. The curve can be acquired by using threejs function catmullromcurve3 to fit the track data to get the 3D spline curve. Then we can get the tangent at every fraction (e.g. const tangent = pointsPath.getTangent(fraction)). However, we don't utilize this method in this work. We calculate the tangent direction from the current point and the next point. As the tangent direction is obtained, the next step is to determine the up vector from the given normal vectors. The function is such as ***up.crossVectors( normal, tangent ).normalize().*** Unfortunately, the tangent direction obtain from the neighboring points is not perfectly orthogonal to the corresponding the normal direction. The tangent is recomputed as ***tangent.crossVectors( up, normal ).normalize()***. Next, the pose of the taxi car at every point is given as following equation.

$$\mathbf{p}^* = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{p} = \begin{bmatrix} 1 & 0 & 0 & a_{14} \\ 0 & 1 & 0 & a_{24} \\ 0 & 0 & 1 & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \tag{1}$$

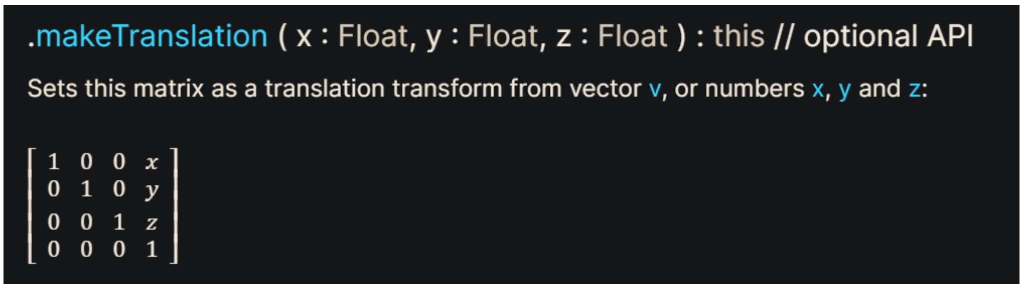


```
FOV: 60
FPS:    17.4
BO_RENDERING
Clipping Near:   0.11
```

```
Current Mesh: TaxiCar
Vertices: 2,045    (202,154)
Faces: 1,252    (152,063)
Selection: v: 0 f: 0
FC WT
```

**Fig. 2** The eagle view of the Mario stadium. The yellow dots are the track and blue arrows indicate the normal direction along the track. The blue axis is z coordinate. A taxi is at the starting line of the stadium.

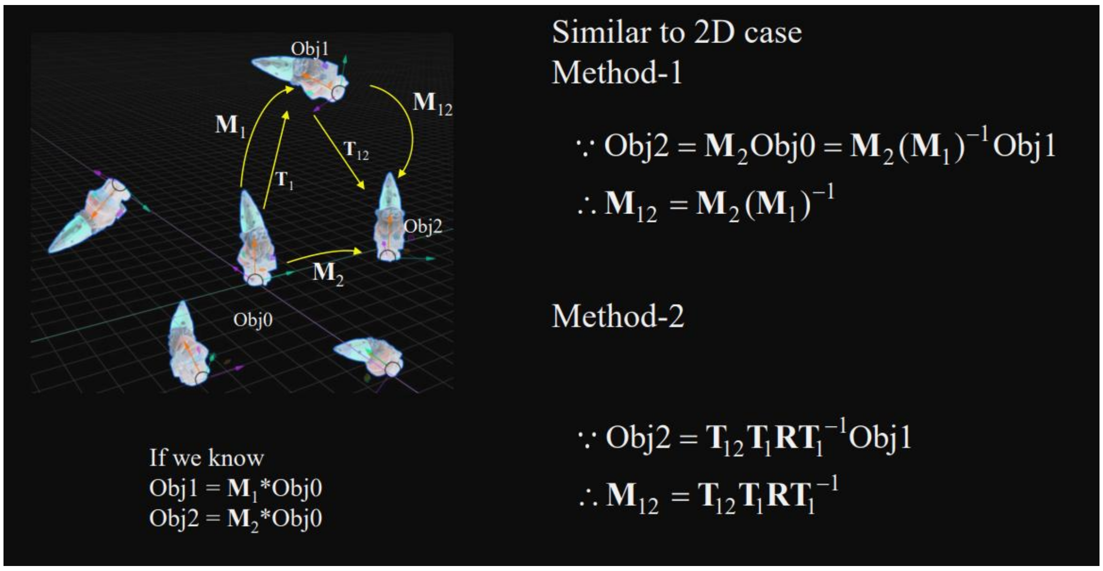
Then, we have to rotate the car at the origin first and then translate it to the next track point position. For the next point, a car is translated back to the origin and reorient itself to its normal axis which is aligned with z-axis. After that, it has to be rotated to match the normal direction of the next point. Once the next orientation is completed, it is been shift toward the next point. The car is moved by the function ***car.position.copy(nextPoint).*** The whole process can be summarized in the Fig.3. The translation function is ***meshCar.applyMatrix4(translationCar)***. The translation function is shown as following



.makeTranslation ( x : Float, y : Float, z : Float ) : this // optional API

Sets this matrix as a translation transform from vector v, or numbers x, y and z:

$$\begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



If we know

$\text{Obj1} = \mathbf{M}_1 * \text{Obj0}$

$\text{Obj2} = \mathbf{M}_2 * \text{Obj0}$

Similar to 2D case

Method-1

$$\because \text{Obj2} = \mathbf{M}_2\text{Obj0} = \mathbf{M}_2(\mathbf{M}_1)^{-1}\text{Obj1}$$

$$\therefore \mathbf{M}_{12} = \mathbf{M}_2(\mathbf{M}_1)^{-1}$$

Method-2

$$\because \text{Obj2} = \mathbf{T}_{12}\mathbf{T}_1\mathbf{R}\mathbf{T}_1^{-1}\text{Obj1}$$

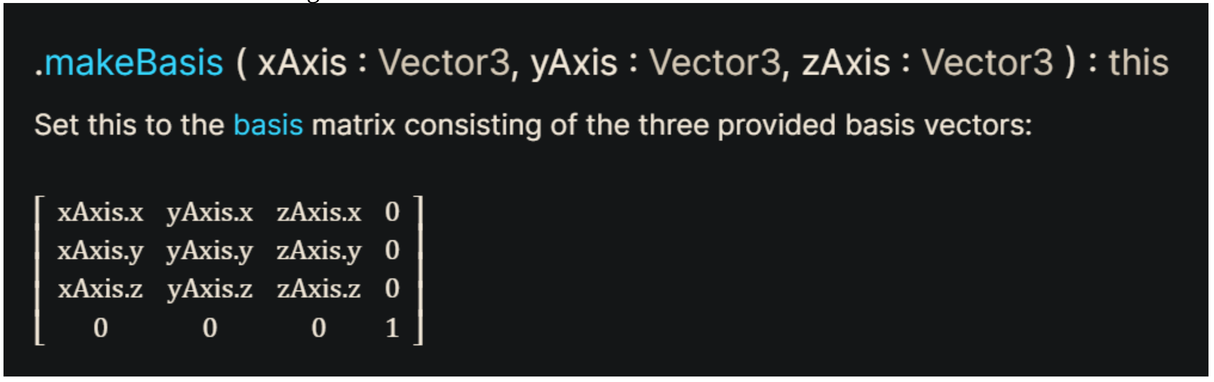
$$\therefore \mathbf{M}_{12} = \mathbf{T}_{12}\mathbf{T}_1\mathbf{R}\mathbf{T}_1^{-1}$$

**Fig. 3** The whole process starting for obj1 is translated back to origin point and then rotate to align to normal axis. Then it has to rotate to new direction, which its normal axis align with the normal vector of the track and move to the new position by a translation from T12.

## Conclusion

A taxi car can run on track at correct orientation at every point on the track. However, we didn't elaborate too much making the car on the right or left lane because the solution is this work is still not perfect enough. A car is offset by a certain amount along its up vector. The drifting is happening around the corner.

When the inverse matrix of the orthogonal matrix is the transpose of the rotation axis. For the translation axis is multiplied by minus one. The function.invert () is to make matrix inverse. The example of the inverse matrix is ***var quatInverse = quat.clone().inverse()***. For building a rotation matrix for the mesh object, the following function ***Matrix4*** is used and is given as



.makeBasis ( xAxis : Vector3, yAxis : Vector3, zAxis : Vector3 ) : this

Set this to the basis matrix consisting of the three provided basis vectors:

$$\begin{bmatrix} xAxis.x & yAxis.x & zAxis.x & 0 \\ xAxis.y & yAxis.y & zAxis.y & 0 \\ xAxis.z & yAxis.z & zAxis.z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## References

[1] 2023 FALL-Advanced Computer Graphics-Lecture-04
[2] https://observablehq.com/@rveciana/three-js-object-moving-object-along-path.
[3] https://threejs.org/docs/#api/en/math/Vector3
[4] https://threejs.org/docs/#api/en/math/Matrix4.makeTranslation