A06 – World and Look-At matrices

This application is a simple car racing game. It is contained in `index.html`, and the procedure for computing and update the world and view matrices are contained in file `WorldView.js`. In particular, a single procedure receives for input the position and the rotation of the car (**`carx`**, **`cary`**, **`carz`**, **`cardir`**) to compute the world matrix, and the position of the camera (**`camx`**, **`camy`**, **`camz`**) to compute the view matrix using the *Look-At* technique described in the course (using for target the position of the car defined by **`carx`**, **`cary`**, **`carz`** and as up-vector the y-axis). Complete the procedure by adding the code to compute the matrices in the appropriate way to make the game playable.

For both the world and view matrices, you can only use library procedures for the basic operations, that are: creating basic rotation and translations, multiplying and inverting matrices, performing vector product and normalize vectors. To this extent, the utils.js library included in project defines the following procedure:

```
utils.crossVector(Va, Vb)
```

that returns the cross product of the two vectors passed in arrays *Va* and *Vb*. And procedure:

```
utils.normalizeVector3(V)
```

that returns a normalized version of its input vector passed in array *V* (i.e. same direction, but unitary length).

Remember that in Javascript array can be created with the following sysntax:

```
[carx, cary, carz]
```

Where for example *carx*, *cary* and *carz* are three variables containing the *x*, *y* and *z* components of the vector. Javascript, however, does not have any internal procedure to sum, subtract, multiply, or do any other operation with vectors.

References:

The algorithm for the third-person camera tracking, and for the car motion, have both been taken from the book:

Game Programming Gems, vol. 4

respectively at chapter 4.1, page 303, and chapter 3.2, page 221. The book can also be found in .pdf format searching on the Web.