cs805 Assignment 1

Shulang Lei
200253624
Department of Computer Science
University of Regina

September 26, 2012

Abstract

This assignment is written in literate programming style, generated by noweb, and rendered by LaTex.

1 Question 1

Let n be a 3 tuple vector, and given that it is along $V1$. It is trivial that we can imply:

$$n = \frac{V1}{[|V1|, |V1|, |V1|]}$$

where $|V1| = \sqrt{V1_x^2 + V1_y^2 + V1_z^2}$

Thus n is now known.

By the definition of cross product, denoted as $\times$ here, knowning that $V1$ and $V2$ is non-collinear, we can also derive:

$$u = \frac{V2 \times V1}{[|V2 \times V1|, |V2 \times V1|, |V2 \times V1|]}$$

Finally, it is also trivial that:

$$v = n \times u$$

2 Question 2

According to the requirement, we need a function that gets the new coordination U, V, N from our two vectors.

First, assuming we have the function already. Thus giving it two points, our function will get the U, V, N from them.

```
<<src/main.cpp>>=
#include <iostream>
#include <typeinfo>//debugging only
#include "util.h"

int main () {
  Point V1;
  decltype(V1) V2;//c11: V2 is of same type of V1

  V1 = {0,0,1};
  V2 = {0,1,1};

  auto uvn = get_uvn(V1, V2);//c11: compiler will replace 'auto' with the right ty

  for (auto point : uvn) {//c11:for each point in uvn
    for (auto num : point) {//c11:for each number in point
      std::cout<<num<<',';
    }
    std::cout<<std::endl;
  }

  return 0;
}
@
```

here is the function.

```
<<src/util.cpp>>=
#include "util.h"
#include <math.h>
```

```
UVN get_uvn(Point V1, Point V2) {

  //get n, which is just normalized V1
  Point n = { V1[0]/get_length(V1),
              V1[1]/get_length(V1),
              V1[2]/get_length(V1) };


  //get u, which is normalized V2 x V1
  Point u_ = { V2[1]*V1[2] - V2[2]*V1[1],
              V2[2]*V1[0] - V2[0]*V1[2],
              V2[0]*V1[1] - V2[1]*V1[0]};
  Point u = { (V2[1]*V1[2] - V2[2]*V1[1])/get_length(u_),
              (V2[2]*V1[0] - V2[0]*V1[2])/get_length(u_),
              (V2[0]*V1[1] - V2[1]*V1[0])/get_length(u_)};

  //get v, which is n x u
  Point v_ = { n[1]*u[2] - n[2]*u[1],
              n[2]*u[0] - n[0]*u[2],
              n[0]*u[1] - n[1]*u[0]};
  Point v = { (n[1]*u[2] - n[2]*u[1])/get_length(v_),
              (n[2]*u[0] - n[0]*u[2])/get_length(v_),
              (n[0]*u[1] - n[1]*u[0])/get_length(v_)};

  UVN result_uvn = {u, v, n};
  return result_uvn;
}

float get_length(Point X) {
  return abs(sqrt(pow(X[0],2)+pow(X[1],2)+pow(X[2],2)));
}
@
```

we need a header file to avoid complicated typedefs.

```
<<src/util.h>>=
#ifndef POINTS_HPP
#define POINTS_HPP
#include <tr1/array>
```

```
typedef std::tr1::array<float, 3> Point;
typedef std::tr1::array<Point, 3> UVN;

UVN get_uvn(Point V1, Point V2);
float get_length(Point X);
#endif
@
```

here is the command to link these files. Notice that I am using -std=c++11 flag to enable c++ 11 features.

```
<<compile.sh>>=
clang++ -std=c++11 -o bin/a.out src/main.cpp src/util.cpp
@
```

Finally, the output binary executable is bin/a.out

3 Question 3