

# NAT: Nostalgic Alien Trespassers — TCR NWERC 2015

Magnus Selin

December 14, 2014

## Contents

|

File: gcd.cpp  
Author: Magnus Selin  
Updated: 2014-12-07  
Import:  
Import:

```
1 int gcd(int a, int b){
2     int t;
3     while (b != 0){
4         t = b;
5         b = a % b;
6         a = t;
7     }
8     return a;
9 }
```

File: vec3.cpp  
Author: Magnus Selin  
Updated: 2014-12-07  
Description: Rational number class  
Import:  
Import: gcd.cpp.

```
1 using namespace std;
2
3 template<class T>
4 class Q{
5 private:
6     T p, q;
7 public:
8     Q(){}
9     Q(T a, T b){
10         p = a; q = b;
11         if(q < 0){p = -p; q = -q;}
12         if(p == 0) q = 1;
13         if(q == 0){
14             // Denominator == 0 -> Handle error
15             q = 1;
16         }
17         T g = gcd(p, q);
18         p /= g; q /= g;
19     }
20     Q operator+(Q a){ return Q(a.p*q + p*a.q, a.q*q); }
21     Q operator-(Q a){ return Q(p*a.q - a.p*q, a.q*q); }
22     Q operator*(Q a){ return Q(a.p*p, a.q*q); }
23     Q operator/(Q a){ return Q(p*a.q, q*a.p); }
24     void operator=(Q a){p = a.p; q = a.q;}
25     bool operator==(Q a){
26         Q f = *this
27         Q s = Q(a.p, a.q);
28         return (f.p == s.p and f.q == s.q);
29     }
30 };
```

File: segment\_segment\_intersection.cpp  
Author: Magnus Selin  
Updated: 2014-12-14  
Import:  
Import:

```
1 typedef vec2<double> vd2;
2 bool segment_segment_intersection(vd2 p, vd2 p1, vd2 q, vd2 q1){
3     vd2 r = p1 - p;
4     vd2 s = q1 - q;
5
6     double t = ((q-p)%s)/(r%s);
7     double u = ((q-p)%r)/(r%s);
```

```
8
9     if(r%s == 0 and (q-p)%r == 0){
10         if((0 <= (q-p)*r and (q-p)*r <= r*r) or
11            (0 <= (p-q)*s and (p-q)*s <= s*s))
12             return true;
13         // Collinear overlapping
14     }
15     else
16         return false;
17     // Collinear not overlapping
18 }
19 else if(r%s == 0 and (q-p)%r != 0){
20     return false;
21     // Parallel
22 }
23 else if(0 <= t and t <= 1 and 0 <= u and u <= 1){
24     return true;
25     // Intersecting at p + tr = q + us
26 }
27 else{
28     return false;
29     // Neither parallel nor do they not intersect
30 }
```

File: three\_points\_to\_circle.cpp  
Author: Magnus Selin  
Updated: 2014-12-11  
Input: Three arbitrary points in 2d space.  
Output: Middle point and radius of the only circle going through the three points.  
Import:  
Import: datastructures/vec3.cpp, datastructures/homogenous\_coord.cpp.

```
1 vector<double> three_points_to_circle(vec3<double> v1, vec3<double> v2,
2     vec3<double> v3){
3     vec3<double> m1 = (v2 - v1)/2 + v1;
4     vec3<double> m2 = (v3 - v2)/2 + v2;
5
6     vec3<double> n1 = vec3<double>(v2[1]-v1[1], -(v2[0]-v1[0]), 1);
7     vec3<double> n2 = vec3<double>(v3[1]-v2[1], -(v3[0]-v2[0]), 1);
8
9     vec3<double> a1 = n1 + m1;
10    vec3<double> a2 = n2 + m2;
11
12    h<double> h1(m1[0],m1[1],1);
13    h<double> h2(a1[0],a1[1],1);
14    h<double> h3(m2[0],m2[1],1);
15    h<double> h4(a2[0],a2[1],1);
16
17    h<double> l1 = h1*h2;
18    h<double> l2 = h3*h4;
19
20    h<double> pm = (l1*l2).norm();
21
22    double r = vec3<double>(pm.x-v1[0], pm.y-v1[1], 0).abs();
23
24    vector<double> ret_val;
25    ret_val.push_back(pm.x);
26    ret_val.push_back(pm.y);
27    ret_val.push_back(r);
28
29    printf("%1f_1%1f_1%1f\n", pm.x, pm.y, r);
30
31    return ret_val;
32 }
```

File: vec3.cpp  
Author: Magnus Selin  
Updated: 2014-12-07  
Description: Class for vectors of arbitrary type.

Import: iostream, cmath.  
Import:

```
1  template <class T>
2  class vec3 {
3  private:
4      T v[3];
5  public:
6      vec3(){}
7      vec3(T a, T b, T c){
8          v[0] = a; v[1] = b; v[2] = c;
9      }
10
11      T operator[] (int i){ return v[i];}
12      vec3 operator=(vec3 o){ v[0] = o[0]; v[1] = o[1]; v[2] = o[2]; }
13      vec3 operator+(vec3 o){
14          return vec3(v[0] + o[0], v[1] + o[1], v[2] + o[2]);
15      }
16      vec3 operator-(vec3 o){
17          return vec3(v[0] - o[0], v[1] - o[1], v[2] - o[2]);
18      }
19      vec3 operator-(){
20          return vec3(-v[0], -v[1], -v[2]);
21      }
22      double operator*(vec3 o){
23          return v[0]*o[0] + v[1]*o[1] + v[2]*o[2];
24      }
25      vec3 operator*(double o){
26          return vec3(v[0]*o, v[1]*o, v[2]*o);
27      }
28      vec3 operator/(double o){
29          return vec3(v[0]/o, v[1]/o, v[2]/o);
30      }
31      vec3 operator%(vec3 o){
32          return vec3(
33              v[1]*o[2] - v[2]*o[1],
34              v[2]*o[0] - v[0]*o[2],
35              v[0]*o[1] - v[1]*o[0]);
36      }
37
38      void operator+=(vec3 o){ *this = *this + o; }
39      void operator-=(vec3 o){ *this = *this - o; }
40      void operator*=(double o){ *this = *this * o; }
41
42      bool operator==(vec3 o){
43          return v[0] == o[0] and v[1] == o[1] and v[2] == o[2];
44      }
45      bool operator!=(vec3 o){
46          return !(*this == o);
47      }
48
49      double abs(){
50          return sqrt(v[0]*v[0] + v[1]*v[1] + v[2]*v[2]);
51      }
52      double ang(vec3 o){
53          if(*this != vec3(0,0,0) and o != vec3(0,0,0))
54              return acos((*this * o) / (this->abs()*o.abs()));
55          return 0;
56      }
57      vec3 norm(){
58          if(*this != vec3(0,0,0))
59              return (*this) / this->abs();
60          return *this;
61      }
62  };
63
64  template<typename T> inline std::ostream &operator<<(std::ostream &os, vec3
```

```
    <T> &v){
65      os << "(" << v[0] << ", " << v[1] << ", " << v[2] << ")";
66      return os;
67  }
```

File: homogenous\_coord.cpp

Author: Magnus Selin

Updated: 2014-12-07

Description: Class handling homogeneous coordinates and lines

Import: cmath.

Import: ../numerical/gcd.cpp.

```
1  template <class T>
2  class h{
3  public:
4      T x, y, z;
5      h(T a, T b, T c){x=a; y=b; z=c;}
6      h(T a, T d1, T b, T d2, T c){x=a*d2; y=b*d1; z=c*d1*d2;}
7      h(){}
8
9      h<T> operator*(h o){ return h<T>(y*o.z-o.y*z, z*o.x-o.z*x, o.x*y-x*o.y)
10         ;}
11
12      h<T> norm() {
13          if(z!=0){
14              if(x/z * z != x or y/z * z != y){
15                  int g = gcd(x, gcd(y,z));
16                  return h<T>(x/g, y/g, z/g);
17              }
18              else return h<T>(x/z, y/z, 1);
19          }
20          return h<T>(1,1,0);
21      }
22  };
```

File: template.cpp

Author: Magnus Selin

Updated: 2014-12-07

Description: Standard template

Import: iostream, cstdlib, cstdio, cmath, vector, set, map, stack, queue, string, bitset, algorithm, cstring.

Import:

```
1  using namespace std;
2
3  #define rep(i, a, b) for(int i = (a); i < int(b); ++i)
4  #define trav(it, v) for(typeof((v).begin()) it = (v).begin(); it != (v).end
5      (); ++it)
6
7  typedef double fl;
8  typedef long long ll;
9  typedef pair<int, int> pii;
10  typedef vector<int> vi;
11
12  bool solve(){
13
14      return true;
15  }
16
17  int main(){
18      int tc=1; //scanf("%d", &tc);
19      rep(i, 0, tc) solve();
20
21      return 0;
22  }
```