L02 Geometry (2D)

1.1 Concepts

1.1.1 Vectors and points

Vector V=<V.x,V.y>: displacement V.x along I axis (horizontal) + V.y along J axis (vertical)
In processing, I goes right, J goes down. Both are one pixel long
||U|| = √(U.x U.x + U.y U.y): magnitude (length) of U, code: n(U)
U•V = U.x V.x + U.y V.y = n(U) n(V) cos(angle(U,V)) (dot/inner product), code: c(U,V)
U×V = –U.x V.y + U.y V.x = n(U) n(V) sin(angle(U,V)) (“cross”/outer product), code: s(U,V)
Quadrants for cos and sin

Point A=(A.x,A.y): location = origin (0,0) + displacement vector <A.x,A.y>
We use (x,y) for points and <x,y> for vectors to help distinguish them
AB = < B.x–A.x , B.y–A.y > vector from A to B, sometimes written B–A, code: V(A,B)
–U = <–U.x,–U.y >, inverse, code: RR(U)
sU = < s U.x, s U.y >, scaled vector, code: S(s,U)
uU+vU : linear combination of vectors, code: W(u,U,v,V)
A+3BC : new point obtained by starting at A and moving 3 times by vector BC, code T(A,3,V(B,C))
$AB^2 = (B.x–A.x)^2 + (B.y–A.y)^2$, distance squared, code d(A,B)
$||AB|| = \sqrt{(AB^2)}$: distance between point A and point B, code: d(A,B)

1.1.2 Properties

B=A+V ⇔ AB = V
BA = –AB
AB+BC = AC
U•V = V•U
U•U = $||U||^2$, denoted U^2
U×V = – V×U
U×U = 0
U•(vV+wW) = v U•V + w U•W
U×(vV+wW) = v U×V + w U×W

1.1.3 Frames

3 points (A,B,C) define a local coordinate system or frame
The local coordinates (x,y) of point P in frame (A,B,C) are defined by P = A + xAB + yAC
Hence AP = xAB + yAC
To obtain x apply ×AC to both sides AP×AC = xAB×AC + yAC×AC, and use AC×AC=0
You get x = AP×AC / AB×AC
Similarly y = AP×AB / AC×AB
Hence, the local coordinates of P are (AP×AC / AB×AC, AP×AB / AC×AB)

1.1.4 Define sets

Vectors orthogonal to N: {V : V•N = 0}
Points on line (Q,T): {P : QP×T = 0}
Points on line passing through A and B: {P : AP×AB = 0}
Points in half-space (Q,N): {P : QP•N ≥ 0}
Points in slab(A,B): {P : AP•AB ≥ 0 && BP•BA ≥ 0 }

1.2 Implementation

1.2.1 Points

```
class pt { float x=0,y=0; pt() {} …} // 2D point class
pt [] P = new pt[5000]; for (int i=0; i<P.length; i++) P[i]=new pt();} // declare an array of points
pt A = P(x,y); // make a new point
float A.x, A.y
A=P[23]; // reference, not a copy! Changing A will change P[23]
show(A);
show(A,r); // disk of radius r and center A
show(A,B); // line segment
show(A,B,C); // triangle
arrow(A,B);  // arrow
label(A,"Here!");
pt Mouse() // current mouse location
pt Pmouse() // previous mouse location
pt ScreenCenter()
drag(A); // translates pt A by mouse displacement: (Mouse() – Pmouse())
float d(A,B) // distance
float d2(A,B) // distance squared
pt A(A,B) // average (A+B)/2 , edge midpoint
pt A(A,B,C) // average (A+B+C)/3 of triangle
pt W(a,A,b,B…) // weighted sum aA+bB+…
pt I(A,s,B) // linear interpolation A+sAB
pt S(A,s,C) // scale A wrt pt C (zoom around fixed point C)
pt R(A,a,C) // rotated version of pt A by angle a around fixed pt C
```

1.2.2 Points and vectors

```
class vec { float x=0,y=0; vec () {};…}
vec U = V(x,y);
float U.x, U.y
vec V(A,B) // vector from A to B, (B–A)
show(A,U); // draw line segment from A to A+U
show(A,s,U); // draw line segment from A to A+sU
Arrow(A,U); // draw arrow from A to A+U
arrow(A,s,U); // draw arrow from A to A+sU
pt T(A,U) // A+U (translation)
pt T(A,s,U) // A+sU (translation)
float n(U) // norm or length of U
vec U(V) // normalized (unit) version of U (same direction, but length 1)
vec U(A,B) // unit vector U(V(A,B))
vec MouseDrag() // V(Pmouse(),Mouse())
vec A(U,V) // average (U+V)/2
vec W(U,V) // sum U+V
vec S(s,U) // scaled vector sU
vec W(u,U,v,V) // weighted sum uU+vV
vec L(U,s,V) // linear interpolation U+s(V–U)
vec I(U,s,V) // interpolate by ratio s from U to V in angle and length (think complex numbers!)
float angle(U,V) // angle from U to V in radians between –π and +π
vec R(U) // U rotated by π/2
vec RR(U) // –U, i.e., U rotated by π
```

vec R(U,a) // U rotated by a
float dot(U,V) // dot product UxVx+UyVy = cos(angle(U,V))*n(U)*n(V)
float c(U,V) // mnemonic for dot product (c=cos)
float cross(U,V) // 2D “cross product” –UxVy+UyVx = sin(angle(U,V))*n(U)*n(V)
float s(U,V) // mnemonic for cross product (c=cos)

1.2.3 Change of frames

You are given a point P and an initial frame (A,B,C). The frame is changed to (Q,R,S). Where is P now? For example, someone has placed 3 fingers on an iPAD at A, B, and C and has moved them to Q, R, and S. They want the whole image to deform accordingly. Implement this space warp.

We say that there is an affine mapping M from (A,B,C) to (Q,R,S) so that M(A)=Q, M(B)=R, and M(C)=S. We want M(P). It may be computed as

```
pt M(pt P, pt A, pt B, pt C, pt Q, pt R, pt S) {
  float x = s(V(A,P),V(A,C))/s(V(A,B),V(A,C));
  float y = s(V(A,P),V(A,B))/s(V(A,C),V(A,B));
  return T(T(Q,x,V(Q,R)),y,V(Q,S)); }
```

Demo

1.2.4 Exercises

Implement Parabola(A,B,C) : {P(t) = (t–1)(t–2)A/2 + t(2–t)B + t(t–1)C/2} and use it to draw the curve
Implement the test and computation of intersection of edge(A,B) with edge(C,D) and test it
Implement a test whether point P lies inside triangle (A,B,C) and test it