```
ON-MOUSE-MOVE(MouseEvent e)
1.  cursorPos ← new Point(e.X, e.Y)
2.  if lensActive
3.      magnifiedBubbleRadius ← DETERMINE-BUBBLE-RADIUS(cursorPos, magTargets, ref currTarget)
4.  else
5.      bubbleRadius ← DETERMINE-BUBBLE-RADIUS(cursorPos, targets, ref currTarget)
6.      if lensReadied == false
7.          TRACK-VELOCITY(cursorPos)
8.          lensReadied ← CHECK-LENS-READY()
9.      else
10.         effectiveSize ← CALCULATE-EFFECTIVE-SIZE(targets, currTarget)
11.         cursorTargetEdgeDistance ← DISTANCE(currTarget, cursorPos) – currTarget.Radius
12.         if (effectiveSize < 10) and (cursorTargetEdgeDistance < 10)
13.             lensActive ← true
14.             magTargets ← GENERATE-MAGNIFIED-TARGETS(cursorPos, targets, currTarget)
```

```
DETERMINE-BUBBLE-RADIUS(Point cursorPos, Targets targets, ref int currentTarget)
1.  for (i = 0; i < targets.Count; i++)
2.      distance ← DISTANCE(cursorPos, targets[i]) – targets[i].Radius
3.      list.ADD(distance, i)
4.  list.SORT()
5.  first ← list[0]
6.  if list > 1
7.      second ← list[1]
8.  else
9.      second ← -1
10. if second ≠ -1
11.     bubbleRadius ← MIN(DISTANCE(cursorPos, targets[first]) + targets[first].Radius,
12.                        DISTANCE(cursorPos, targets[second]) - targets[second].Radius)
13. else
14.     bubbleRadius ←DISTANCE(cursorPos, targets[first]) + targets[first].Radius
15. currentTarget ← first
16. return bubbleRadius
```

```
TRACK-VELOCITY(Point cursorPos)
1.  timePoint ← new TimePoint(cursorPos.X, cursorPos.Y, Time.NowMS)
2.  moves.ADD(timePoint)
3.  resampledPosition ← RESAMPLE-IN-TIME(moves, hertz)
4.  resampledVelocity ← DERIVATIVE(resampledPosition)
5.  smoothedVelocity ← FILTER(resampledVelocity, kernel)
6.  maxima.CLEAR()
7.  localMaxima ← SERIES-MAXIMA(smoothedVelocity, 0, -1)
8.  for (i = 0; i < localMaxima.Length; i++)
9.      maxima.ADD(new TimePoint(resampledPosition[localMaxima[i]].X,
10.                              resampledPosition[localMaxima[i]].Y,
11.                              resampledPosition[localMaxima[i]].Time – resampledPosition[0].Time))
12. minima.CLEAR()
13. localMinima ← SERIES-MINIMA(smoothedVelocity, 0, -1)
```

```
14.     for (i = 0; i < localMinima.Length; i++)
15.         minima.ADD(new TimePoint(resampledPosition[localMinima[i]].X,
16.                                  resampledPosition[localMinima[i]].Y,
17.                                  resampledPosition[localMinima[i]].Time – resampledPosition[0].Time))
```

CHECK-LENS-READY()

```
1.      lensReady ← false
2.      if maxima.Count > 1
3.          if maxima.Count ≥ ballisticPhase
4.              lensReady ← true
5.          else if ((ABS(maxima[0].X – minima[0].X) < 20)
6.              ballisticPhase = 3
7.          else
8.              lensReady ← true
9.      return lensReady
```

CALCULATE-EFFECTIVE-SIZE(Targets *targets*, int *currentTarget*)

```
1.      min ← ∞
2.      for (i = 0; i < targets.Count; i++)
3.          if i ≠ currentTarget
4.              edgeDistance ← DISTANCE(targets[i], targets[currentTarget]) –
5.                                  targets[i].Radius – targets[currentTarget].Radius
6.              if edgeDistance < min
7.                  min ← edgeDistance
8.      effectiveSize ← (targets[currentTarget].Size) + (min / 2)
9.      return effectiveSize
```

GENERATE-MAGNIFIED-TARGETS(Point *cursorPos*, Targets *targets*, int *currentTarget*)

```
1.      for (i = 0; i < targets.Count; i++)
2.          magTargets[i] ← new Target
3.          magIndexes[i] ← 0
4.      sortedTargets = SORT-BY-EDGE-DISTANCE(targets, currentTarget)
5.      for (i = sortedTargets.Count – 1; i ≥ 0; i--)
6.          unMagIndexes.PUSH(sortedTargets[i])
7.      while unMagIndexes is not empty
8.          nextTarget ← targets[unMagIndexes.Top]
9.          closestReferenceIndex ← FIND-CLOSEST-REFERENCE-TARGET(magIndexes, targets, nextTarget)
10.         newMagTarget ← GENERATE-MAGNIFIED-TARGET(targets[closestReferenceIndex],
11.                                  magTargets[closestReferenceIndex], nextTarget)
12.         magIndexes[unMagIndexes.Top] ← 1
13.         magTargets[unMagIndexes.Top] ← newMagTarget
14.         unMagIndexes.POP()
15.     return magTargets
```

```
SORT-BY-EDGE-DISTANCE(Targets targets, int currentTarget)
1.    for (i = 0; i < targets.Count; i++)
2.        if (i ≠ currentTarget)
3.            edgeDistance ← DISTANCE(targets[i], targets[currentTarget]) –
4.                                    targets[i].Radius – targets[currentTarget].Radius
5.            sortedList.ADD(edgeDistance, i)
6.    sortedList.SORT()
7.    return sortedList
```

```
FIND-CLOSEST-REFERENCE-TARGET(List magnifiedIndexes, Targets targets, Target currentTarget)
1.    min ← ∞
2.    for (i = 0; i < magnifiedIndexes.Count; i++)
3.        if (magnifiedIndexes[i] == 1)
4.            edgeDistance ← DISTANCE(targets[i], targets[currentTarget]) –
5.                                    targets[i].Radius – targets[currentTarget].Radius
6.            if edgeDistance < min
7.                minIndex ← i
8.                min ←edgeDistance
9.    return minIndex
```

```
GENERATE-MAGNIFIED-TARGET(Target referenceTarget, Target magReferenceTarget, Target nextTarget)
1.    angleBetweenOrig ← ANGLE(nextTarget, referenceTarget)
2.    angleBetweenMag ← ANGLE(nextTarget, magReferenceTarget)
3.    rotateAngleOrig ← 2π – angleBetweenOrig
4.    rotateAngleMag ← 2π – angleBetweenMag
5.    rotatedPointOrig ← ROTATE(nextTarget, referenceTarget, rotateAngleOrig)
6.    rotatedPointMag ← ROTATE(nextTarget, magReferenceTarget, rotateAngleMag)
7.    edgeDistance ← (referenceTarget.X – referenceTarget.Radius) –
8.                    (rotatedPointOrig.X + nextTarget.Radius)
9.    rotatedPointMag.X ← (magReferenceTarget.X – magReferenceTarget.Radius) –
10.                       (edgeDistance + (nextTarget.Radius × magFactor))
11.   rotatedPointMag ← ROTATE(rotatedPointMag, magReferenceTarget, –rotateAngleOrig)
12.   return new Target(rotatedPointMag, nextTarget × magFactor)
```

Publication

Mott, M.E. and Wobbrock, J.O. (2014). Beating the bubble: Using kinematic triggering in the Bubble Lens for acquiring small, dense targets. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '14).* Toronto, Ontario (April 26-May 1, 2014). New York: ACM Press, pp. 733-742.

Acknowledgement

This work was supported in part by the National Science Foundation under grant IIS-0952786. Any opinions, findings, conclusions or recommendations expressed in this work are those of the authors and do not necessarily reflect those of the National Science Foundation.