Node.moveQ()

ref: SEEK

AttemptAcceptFromInputs()

```
[serielizedfield] int lastInpIndex = 0;
//................................//

void AttemptAcceptFromInputs()
{
    for(int i0 = 1; i0 <= INP.Count; i0 += 1)
        int index = (lastInpIndex + i0) % INP.Count
        Node pred = INP[index];

        if(pred.Q.Count == 0)
            continue

        float pred_FutureHeadDist = pred.Q.First.Dist + speed * dt

        if(pred_FutureHeadDist >= 1f)
            if((this.Q.Count == 0) || (this.Q.Tail.Dist - pred_FutureHeadDist % 1f >= minSpacing))
                //
                Item head = pred.Dequeue()
                head.Dist = head.Dist % 1f; UpdateTransform(head);
                this.Q.Enqueue(head)
                //
                this.lastInpIndex = index
                return

    // If none could send, do nothing this frame (and next frame we start from lastInpIndex + 1 again)

}
```

AttemptSendToSuccessor()

```
[serielizedfield] int lastOutIndex = 0;
//................................//

void AttemptSendToSuccessor()
{
    if(this.Q.Count == 0)
        return;

    float our_FutureHeadDist = this.Q.First.dist + speed * dt;

    for(int i0 = 1; i0 <= OUT.Count; i0 += 1)
        int index = (lastOutIndex + i0) % OUT.Count
        Node succ = OUT[index]

        if(our_FutureHeadDist >= 1f) // could be done before loop
            if(succ.Q.Head.Dist - our_FutureHeadDist % 1f >= minSpacing)
                //
                Item head = this.Dequeue()
                head.Dist = head.Dist % 1f; UpdateTransform(head);
                succ.Q.Enqueue(head)
                //
                this.lastOutIndex = index
                return
    // If none could recieve, do nothing this frame (and next frame we start from lastOutIndex + 1 again)


    this.Q.First.Dist = C.clamp(our_FutureHeadDist, e, 1f - e);
    UpdateTransform(Q.First)
}
```

SlideRemainingItems()

```
void SlideRemainingItems()
{
    if(this.Q.Count <= 1)
        return // either empty or single-already handled in AttemptSendToSuccessor()

    for(int i0 = 1; i0 < Q.Count; i0 += 1)
        float item_FutureDist =   Q[i0].Dist + speed * dt;

        if(Q[i0 - 1].Dist - item_FutureDist >= minSpacing)
            Q[i0].Dist = C.clamp(item_FutureDist, e, 1f - e );
            UpdateTransform(Q[i0])
}
```

moveQ()

```
void moveQ()
{
    // 1. first attempt to move to successor
    AttemptSendToSuccessor()

    // 2. next attempt to slide items other than head
    SlideRemainingItems()

    // 3. next attempt to take from predecessor
    AttemptAcceptFromInputs()
}
```

Node

```
public class Node
{
    public int id;
    public HashSet<Node> INP, OUT;
    public HashSet<Node> Q;
}
```