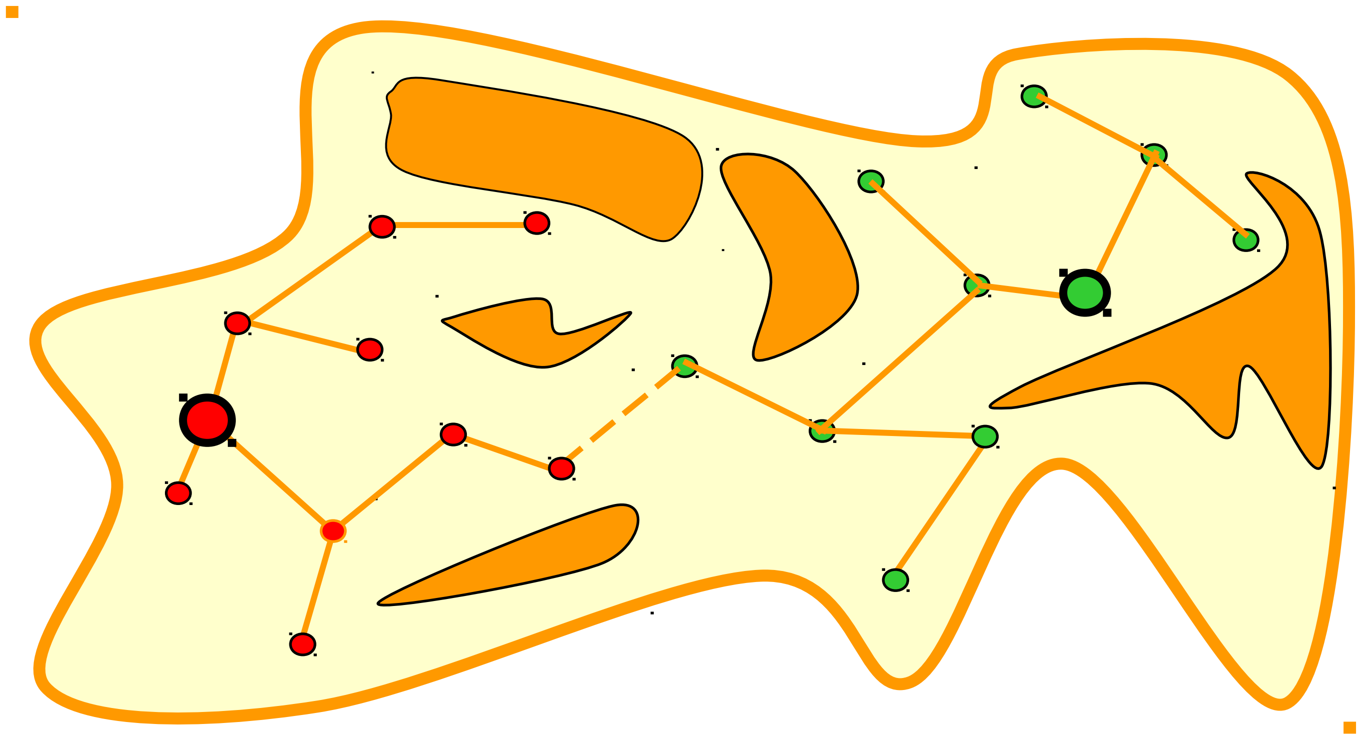
Probabilistic roadmaps (PRMs)



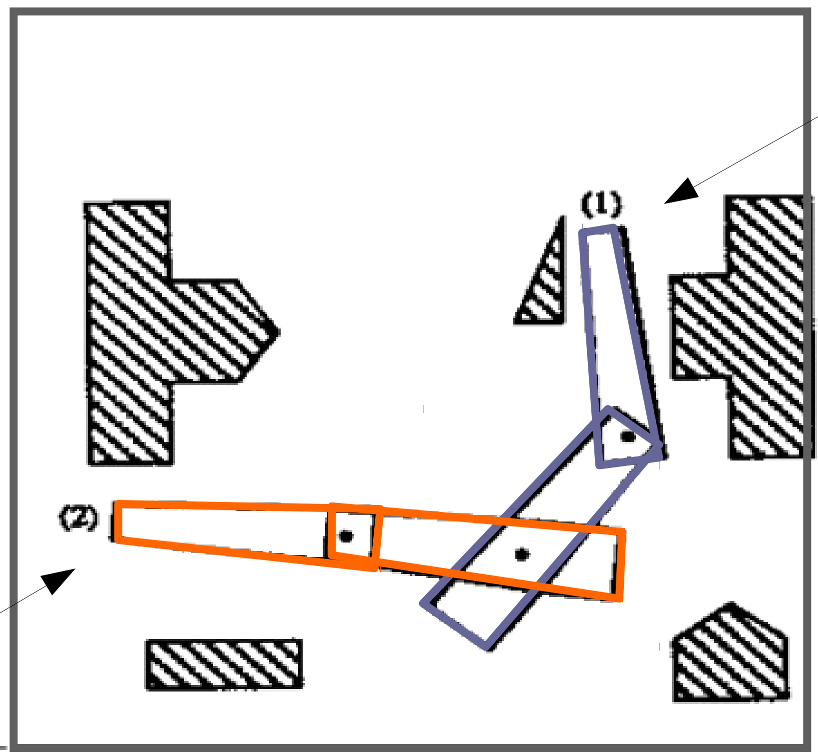
How do you plan in high dimensional state spaces?

Problem we want to solve

<u>Given</u>:
– a point-robot (robot is a point in space)
– a start and goal configuration

<u>Find</u>:
– path from start to goal that does not result in a collision

Starting configuration



Goal configuration

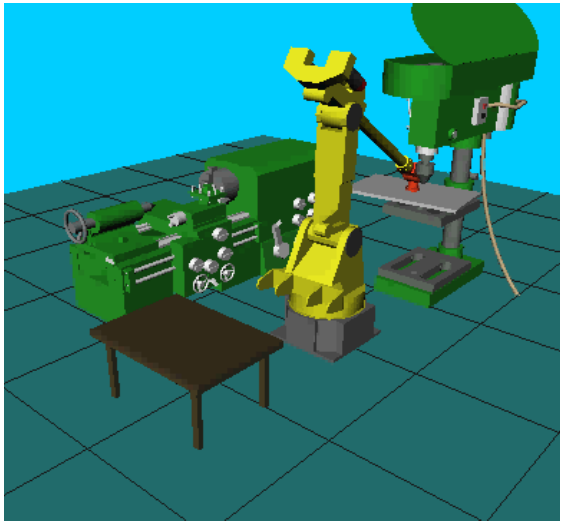
Problem we want to solve

<u>Given</u>:
– a point-robot (robot is a point in space)
– a start and goal configuration

<u>Find</u>:
– path from start to goal that does not result in a collision

<u>Assumptions</u>:
– the position of the robot can always be measured perfectly
– the motion of the robot can always be controlled perfectly
– the robot can move in any directly instantaneously

<u>For example:</u> think about a robot workcell in a factory...



Probabilistic roadmaps (PRMs)

PRMs are specifically designed for high-dimensional configuration spaces – such as the c-space of a robot arm

<u>Problem:</u> robot arm configuration spaces are typically high dimensional

– for example, imagine using the wavefront planner to solve a problem w/ a 10-joint arm

– several variants of the path planning problem have been proven to be PSPACE-hard.

Probabilistic roadmaps (PRMs)

PRMs are specifically designed for high-dimensional configuration spaces – such as the c-space of a robot arm

General idea:

– create a randomized algorithm that will find a solution quickly in many cases

– eventually, the algorithm will be guaranteed to find a solution if one exists *with probability one*

Probabilistic roadmaps (PRMs)

PRMs are specifically designed for high-dimensional configuration spaces – such as the c-space of a robot arm

<u>General idea:</u>

– create a randomized algorithm that will find a solution quickly in many cases

– but, eventually, the algorithm will be guaranteed to find a solution if one exists *with probability one*

<u>With probability one --> “Almost surely”</u>
– the probably of an event NOT happening approaches zero as the algorithm continues to run

Example: an infinite sequence of coin flips contains at least one tail almost surely.

Probabilistic roadmaps (PRMs)

PRM
– su

Gen

– cr
case

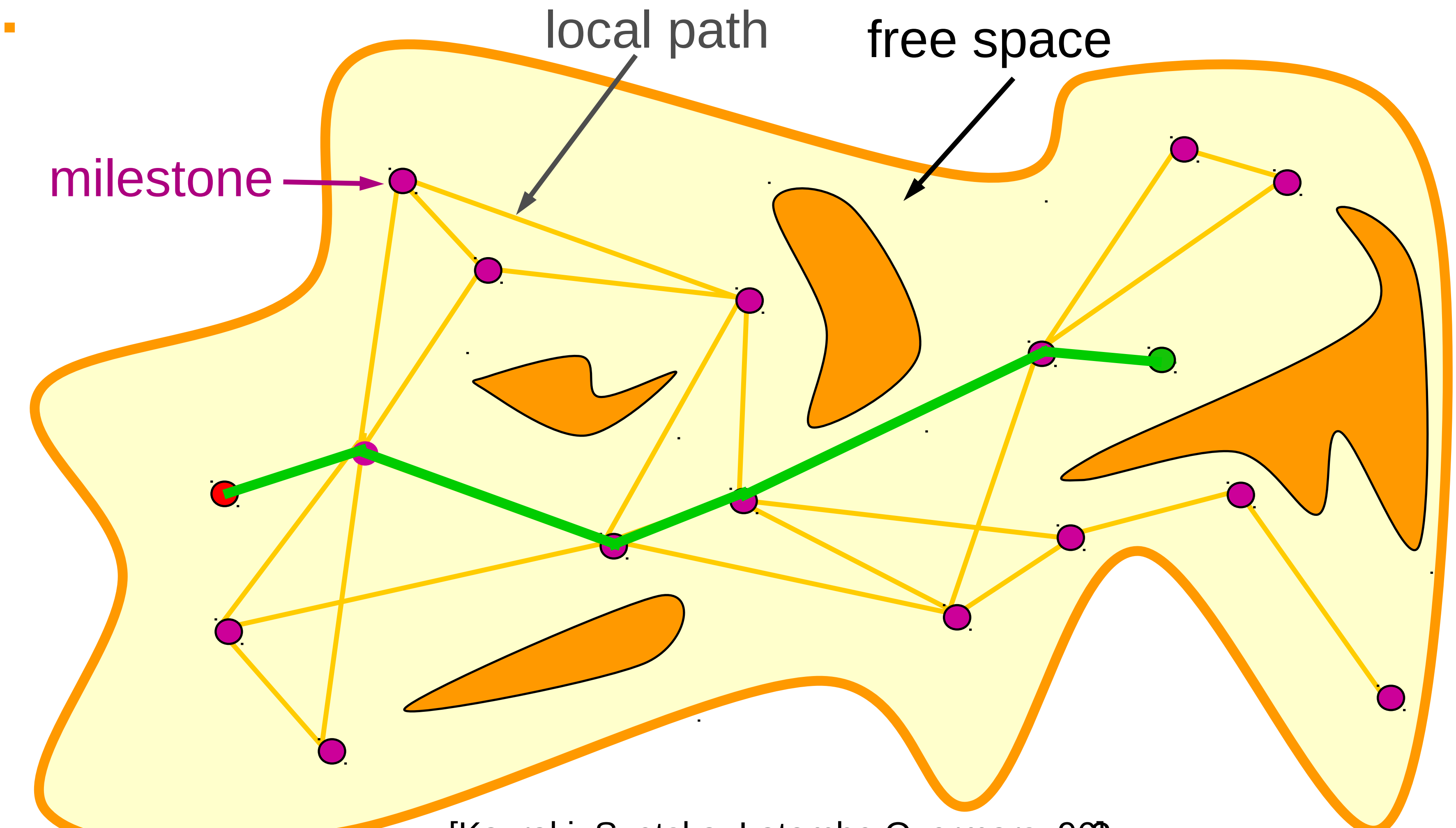
– bu
exis

Infinite monkey theorem:

A monkey typing keys randomly on a keyboard will produce any given text (the works of William Shakespeare) *almost surely*

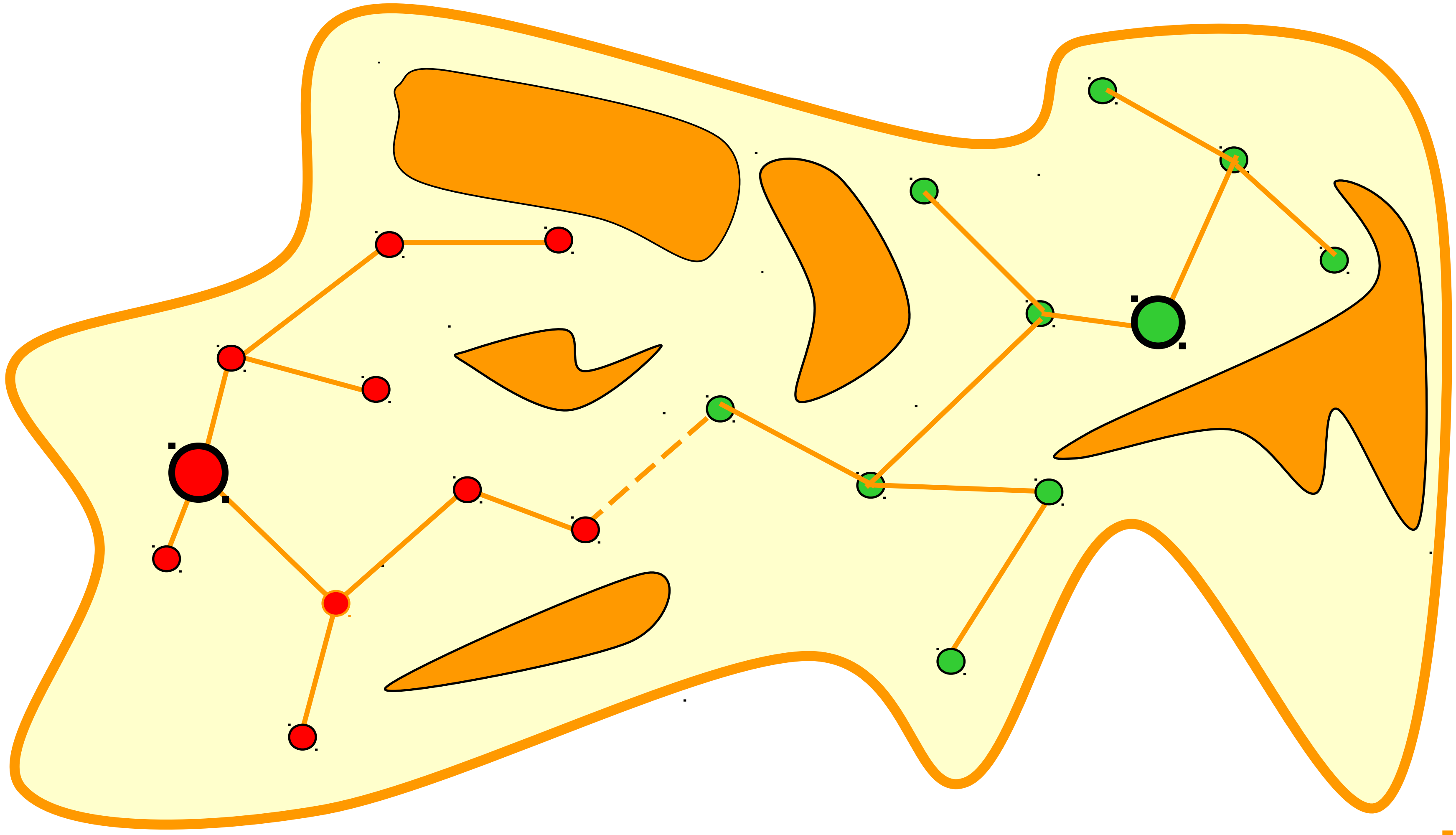
Example: an infinite sequence of coin flips contains at least one tail almost surely.

Probabilistic Roadmap (PRM): multiple queries



[Kavraki, Svetska, Latombe,Overmars, 96]

Probabilistic Roadmap (PRM): single query



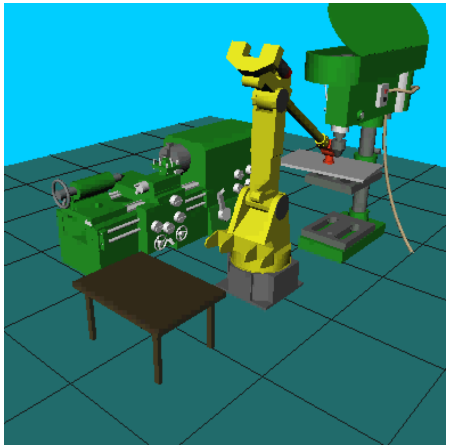
Multiple-Query PRM

Classic multiple-query PRM

- *Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces,* L. Kavraki *et al*., 1996.

Assumptions

- Static obstacles
- Many queries to be processed in the same environment
- Examples
  - Navigation in static virtual environments
  - Robot manipulator arm in a workcell



Overview

- Precomputation: roadmap construction
  - Uniform sampling
  - Resampling (expansion)
- Query processing

Uniform sampling

Input: geometry of the robot & obstacles

Output: roadmap G = (V, E)

```
V ← ∅ and E ← ∅.
repeat
  q ← a configuration sampled uniformly at random from C.
  if CLEAR(q)then
    Add q to V.
    Nq ← a set of nodes in V that are close to q.
    for each q'∈ Nq, in order of increasing d(q,q')
      if LINK(q',q)then
        Add an edge between q and q' to E.
```

Some terminology

- The graph G is called a **probabilistic roadmap**.
- The nodes in G are called **milestones**.

Query processing

- Connect q_{init} and q_{goal} to the roadmap
- Start at q_{init} and q_{goal}, perform a random walk, and try to connect with one of the milestones nearby
- Try multiple times

Error

- If a path is returned, the answer is always correct.
- If no path is found, the answer may or may not be correct. We hope it is correct with high probability.

Probabilistic completeness of PRM

Theorem (Kavraki et al 1998):

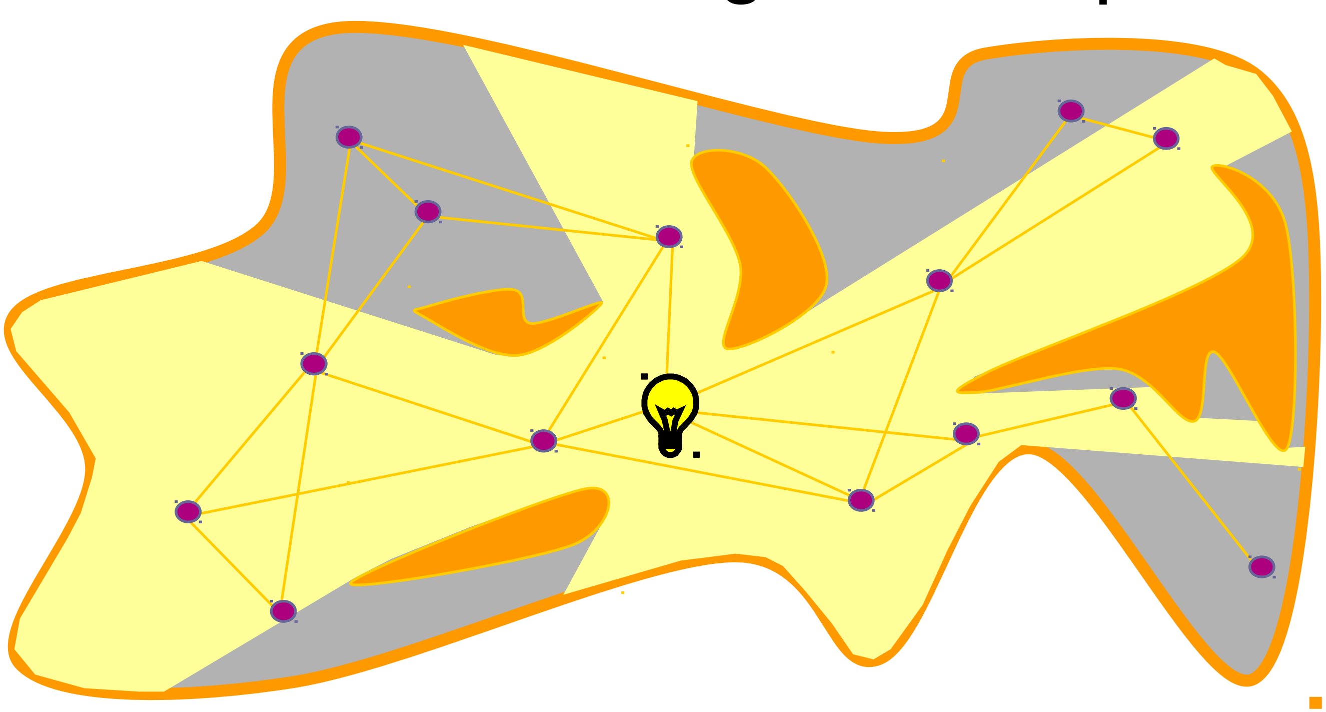
If a path planning problem is feasible, then there exist constants n_0 and a>0, such that:

$$P(\text{a path is found}) \geq 1 - e^{-an}$$

where *n*>n_0 is the number of samples

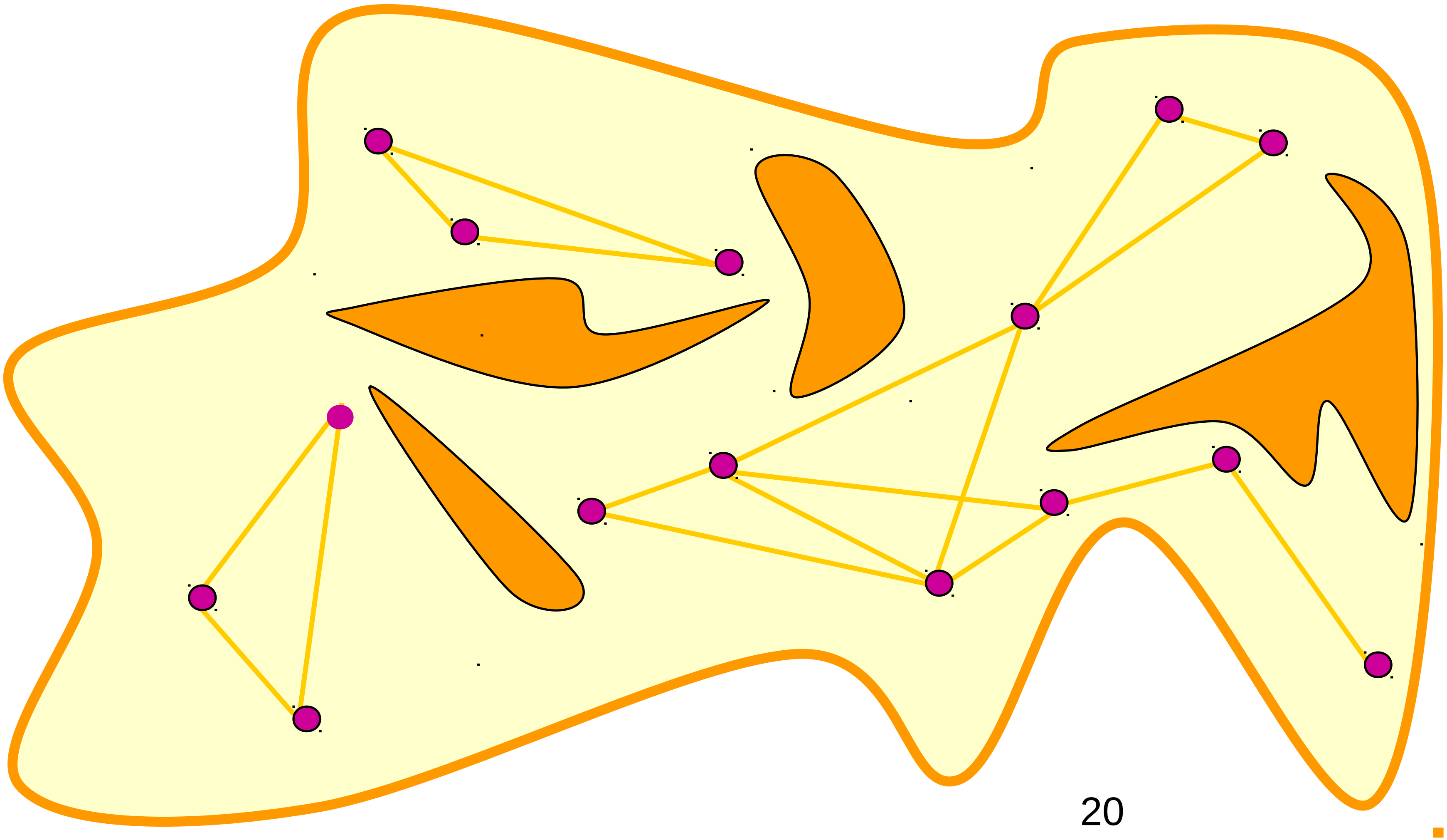
Why does it work? Intuition

- A small number of milestones **almost** "cover" the **entire** configuration space.



Difficulty

- Many small connected components



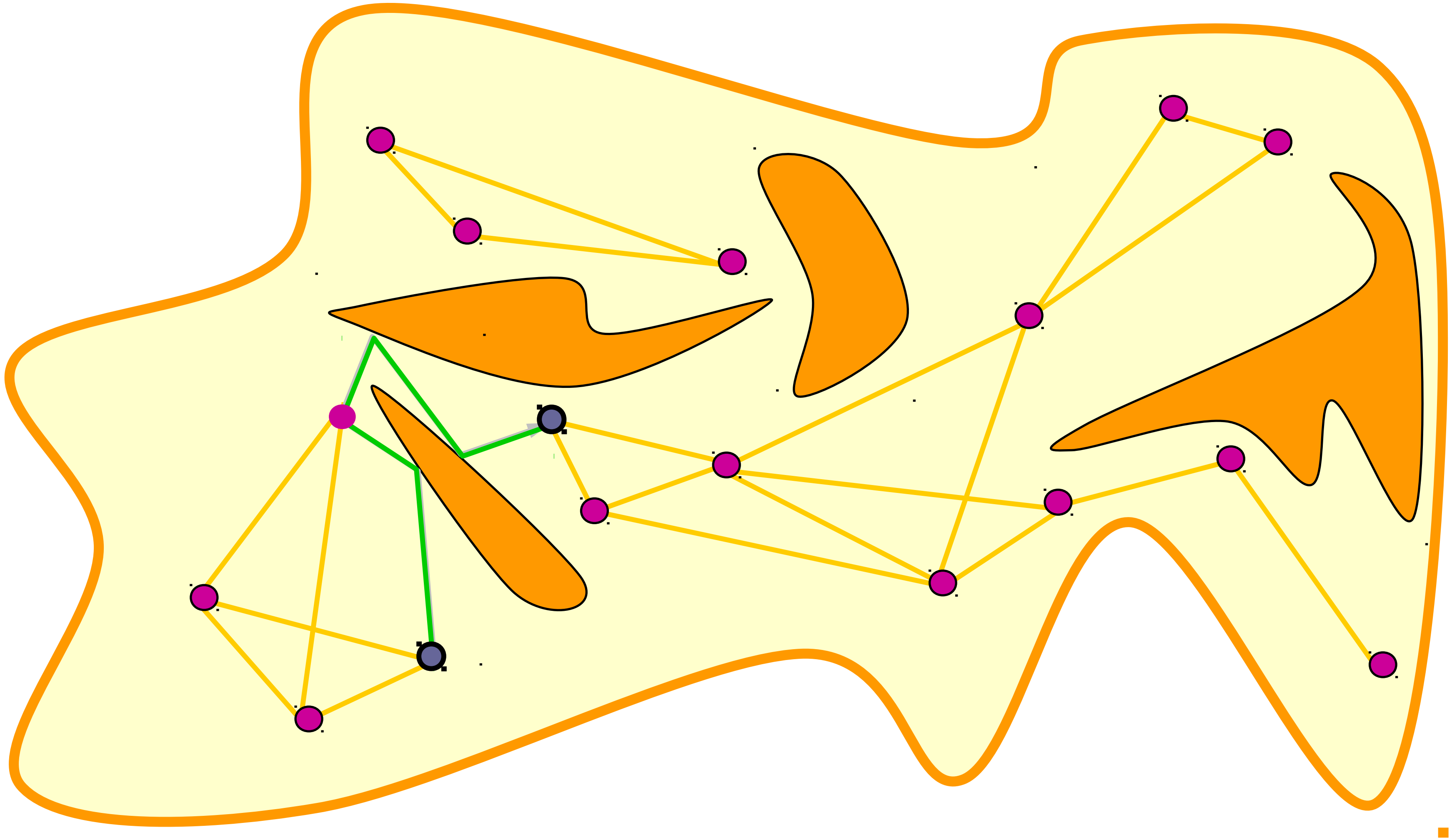
Resampling (expansion)

- Failure rate $$r(q) = \frac{\text{no. failed LINK}}{\text{no. LINK}}$$

- Weight $$w(q) = \frac{r(q)}{\sum_p r(p)}$$

- Resampling probability $$\Pr(q) = w(q)$$

Resampling (expansion)



Resampling (expansion)

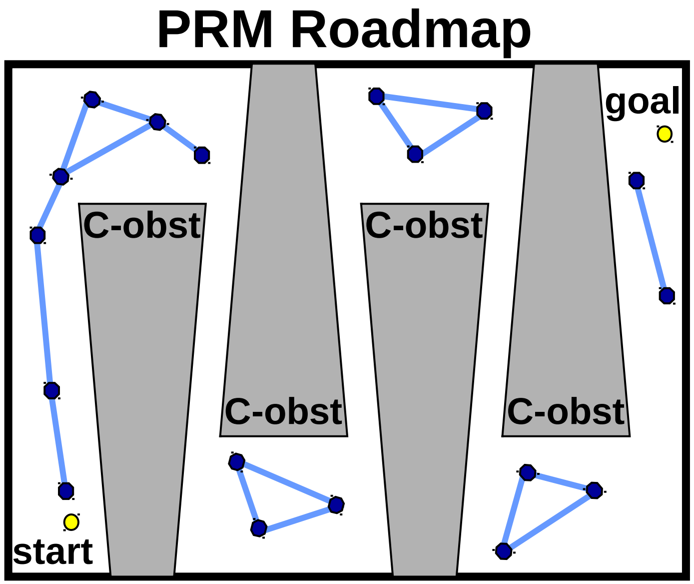
Once a node is selected to be expanded:

1. Pick a random motion direction in c-space and move in this direction until an obstacle is hit.
2. When a collision occurs, choose a new random direction and proceed for some distance.
3. Add the resulting nodes and edges to the tree. Re-run tree connection step.

Gaussian sampler

So far, we have only discussed uniform sampling...

Problem: uniform sampling is not a great way to find paths through narrow passageways.

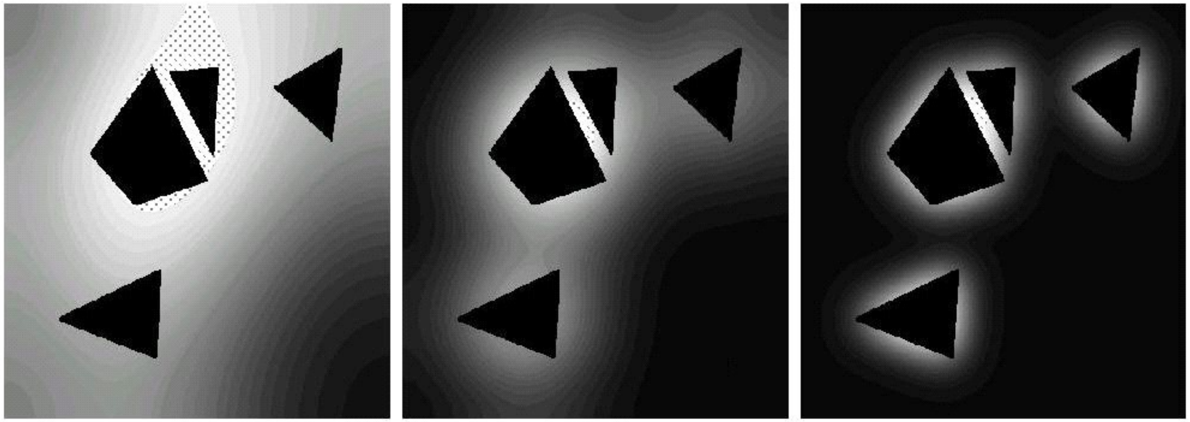


Gaussian sampler

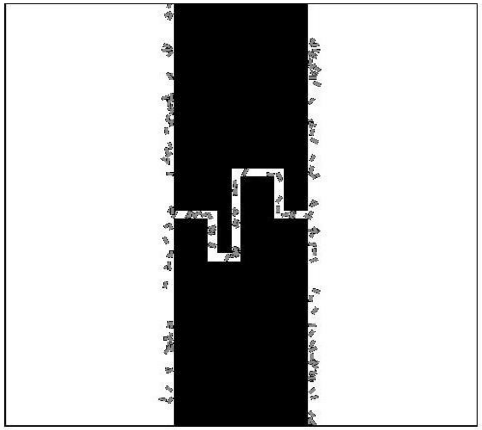
<u>Gaussian sampler:</u>

– Sample points uniformly at random (as before)
– For each sampled point, sample a second point from a Gaussian distribution centered at the first sampled point
– Discard the first sample if both samples are either free or in collision
– Keep the fist sample if the two samples are NOT both free or both in collision (that is, keep the sample if the free/collision status of the second sample is different from the first).

Gaussian sampler



Probability of sampling a point under the Gaussian sampler as a function of distance from a c-space obstacle



Example of samples drawn from Gaussian sampler

Single-Query PRM

Lazy PRM

- *Path Planning Using Lazy PRM*, R. Bohlin & L. Kavraki, 2000.

Precomputation: roadmap construction

- Nodes
  - Randomly chosen configurations, which may or may **not** be collision-free
  - No call to **CLEAR**
- Edges
  - an edge between two nodes if the corresponding configurations are close according to a suitable metric
  - no call to **LINK**

Query processing: overview

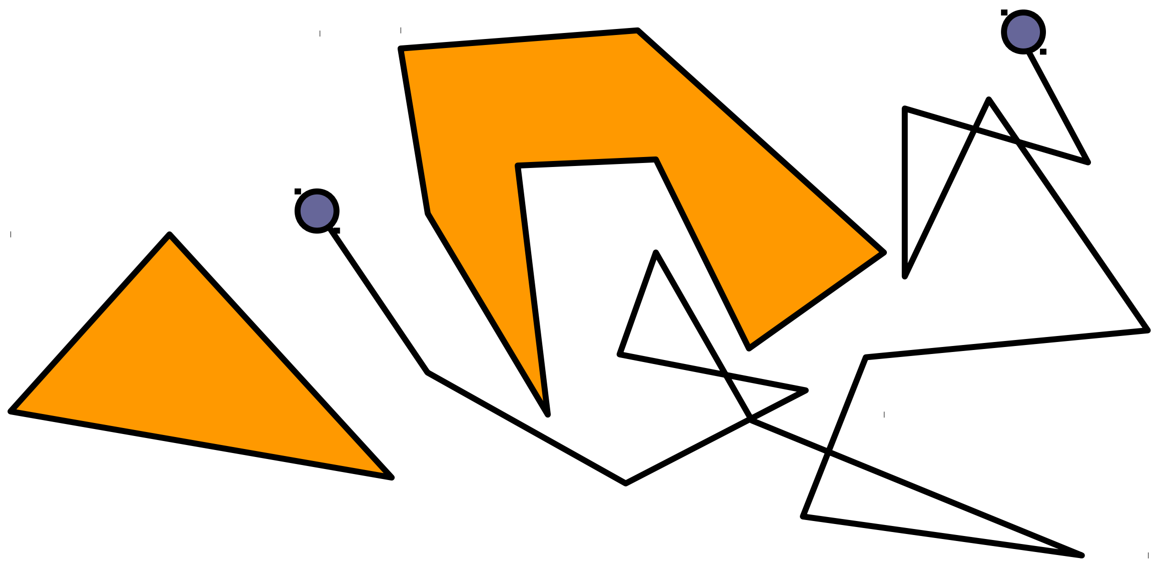
1. Find a shortest path in the roadmap
2. Check whether the nodes and edges in the path are collision.
3. If yes, then done. Otherwise, remove the nodes or edges in violation. Go to (1).

We either find a collision-free path, or exhaust all paths in the roadmap and declare failure.

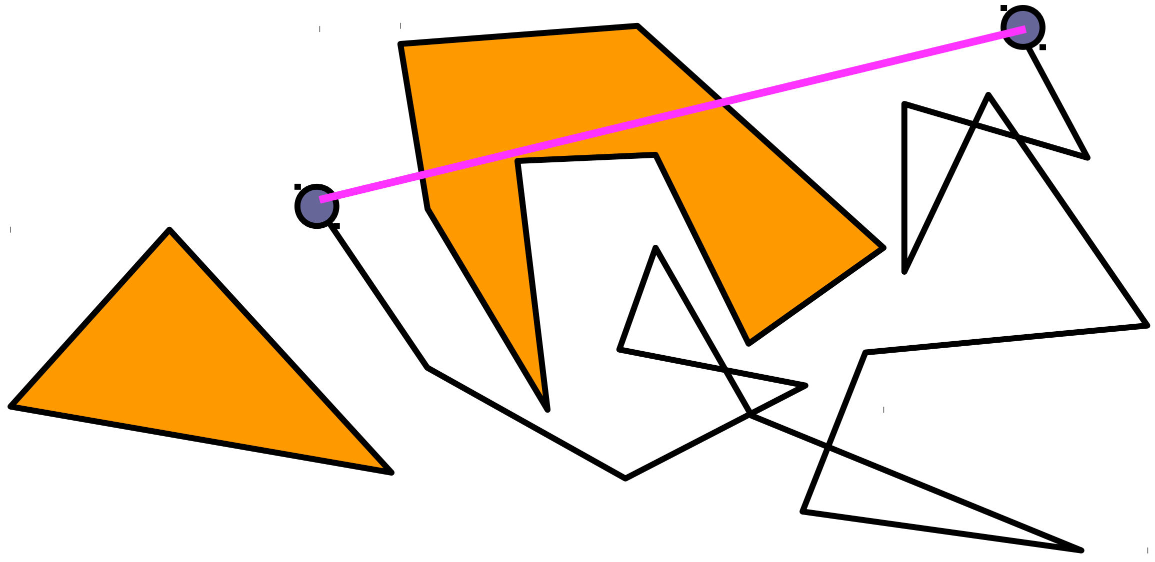
Query processing: details

- Find the shortest path in the roadmap
  - A* algorithm
  - Dijkstra's algorithm (uniform cost search)
- Check whether nodes and edges are collisions free
  - **CLEAR**(q)
  - **LINK**(q_0, q_1)

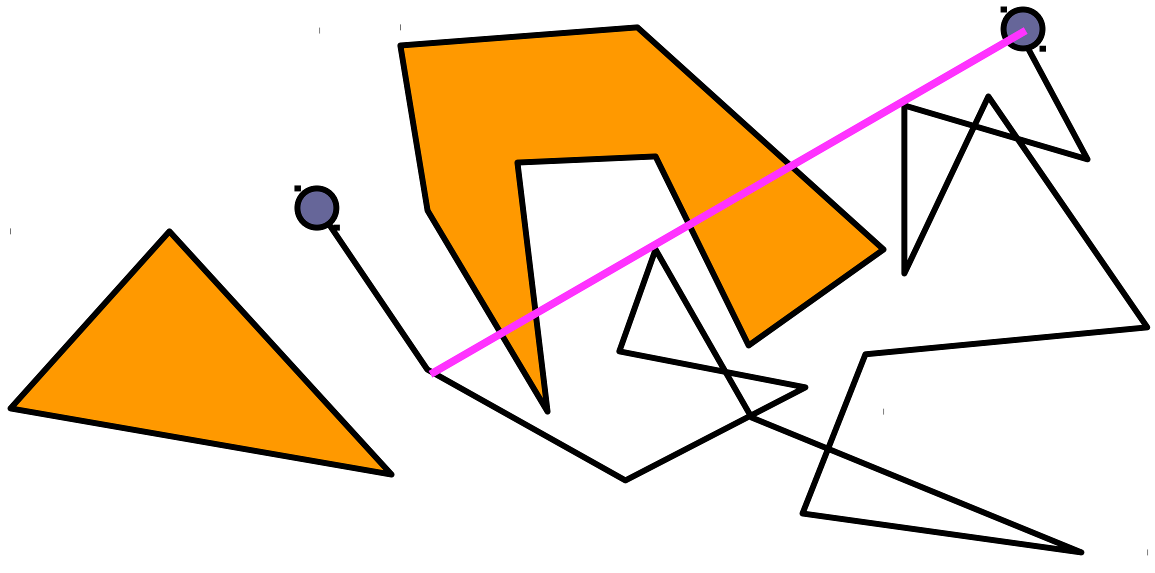
Smoothing the path



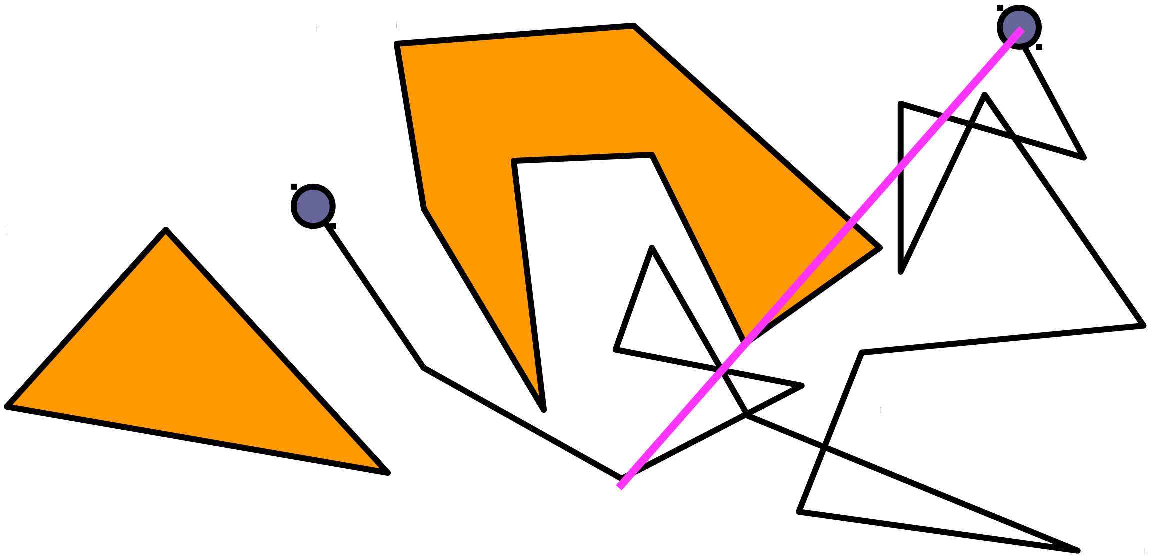
Smoothing the path



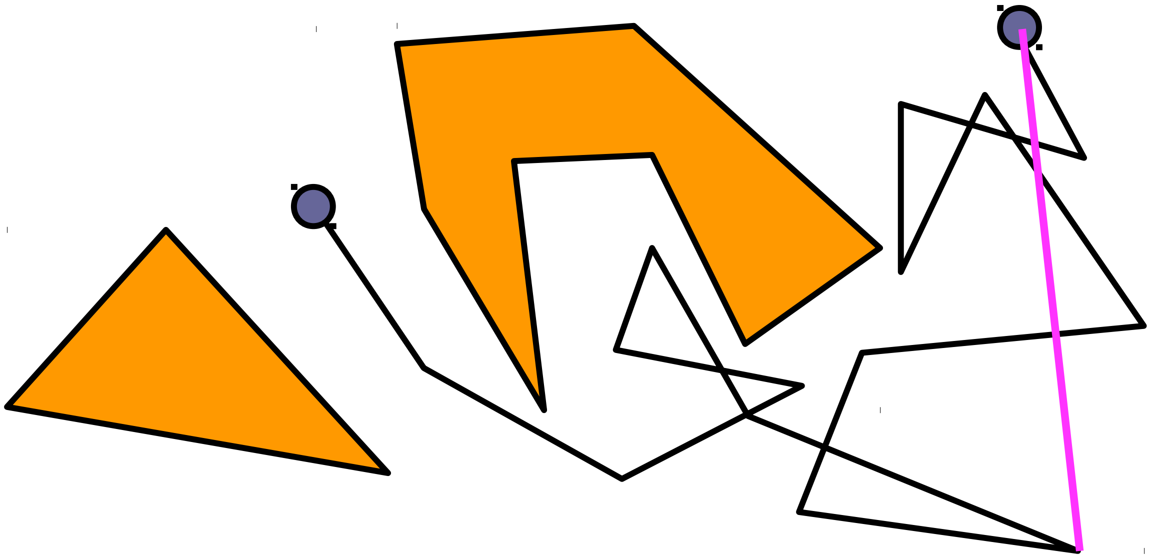
Smoothing the path



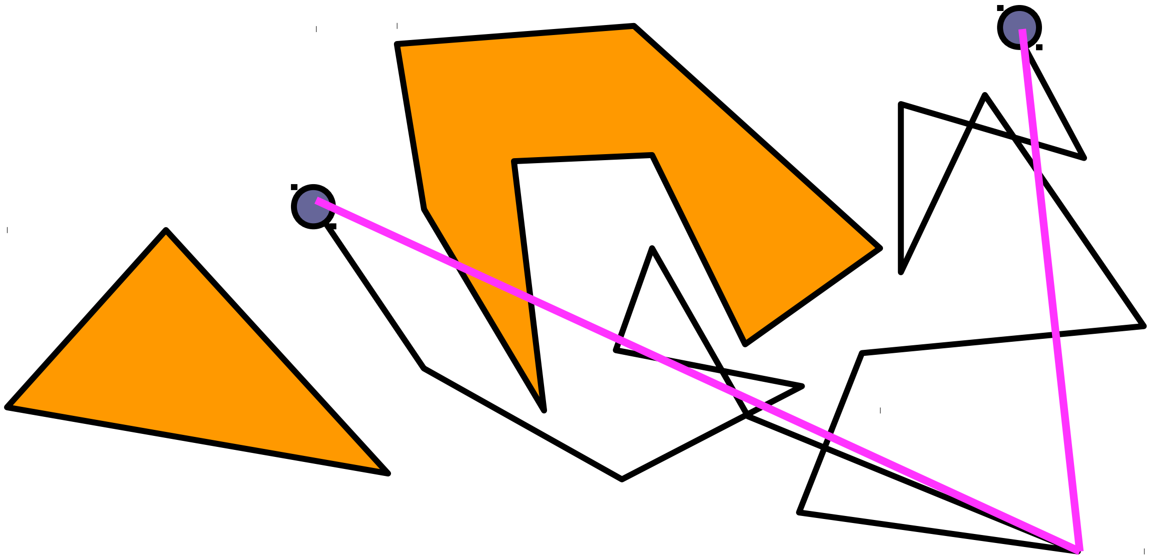
Smoothing the path



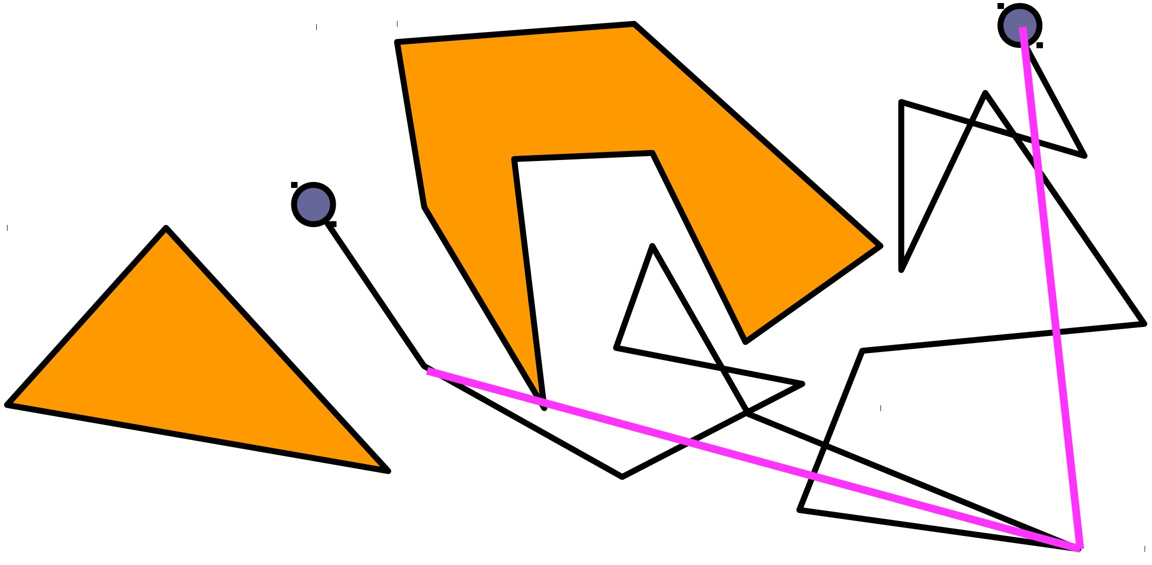
Smoothing the path



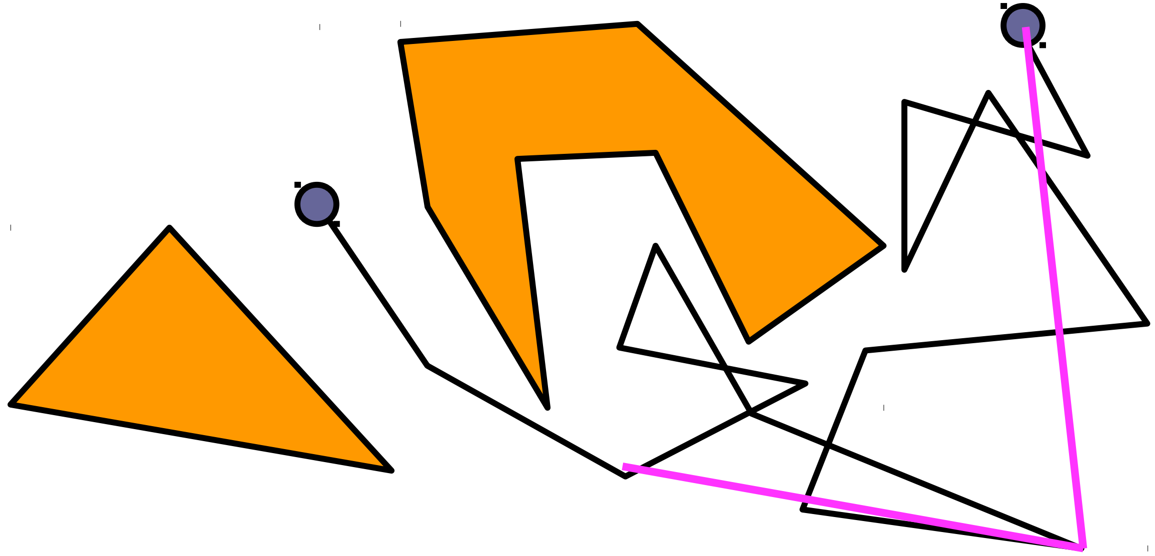
Smoothing the path



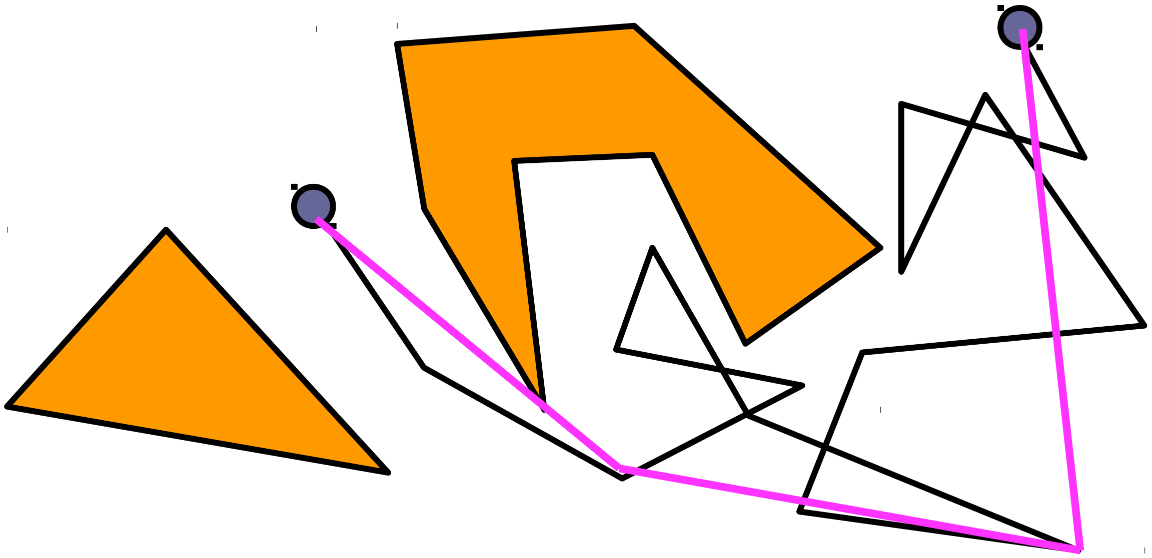
Smoothing the path



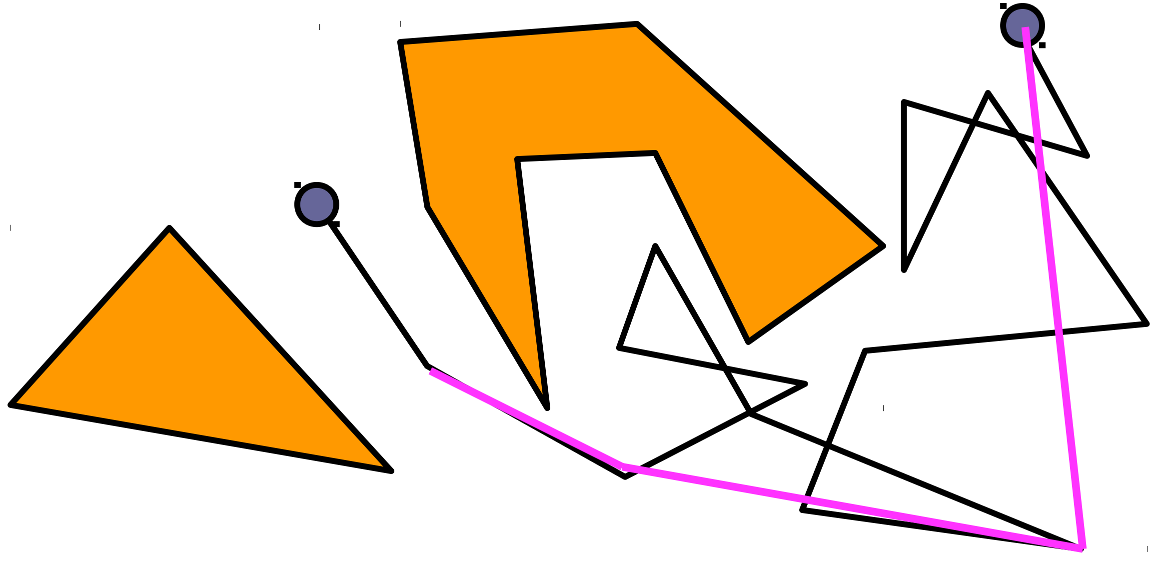
Smoothing the path



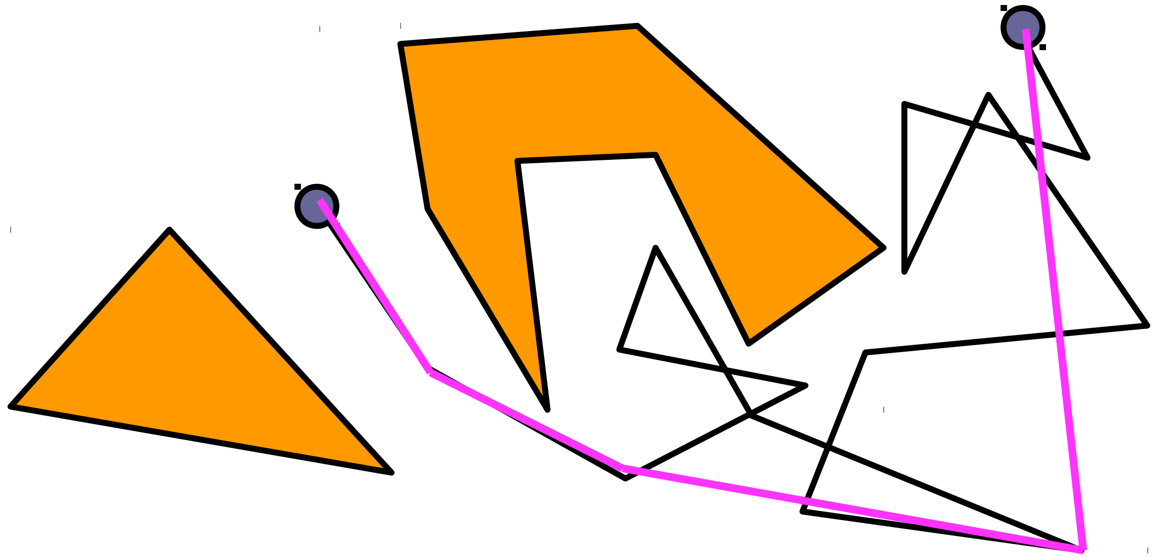
Smoothing the path



Smoothing the path



Smoothing the path



Smoothing the path