CS 460: INTRO TO COMPUTATIONAL ROBOTICS

Assignment 1

Andy Xu
axx2@rutgers.edu
Tasha Pais
tdp74@rutgers.edu
Rutgers University, Department of Computer Science
October 8, 2023

1 Generate, Visualize and Store 2D Polygonal Scenes

Examples and Configurations

The first 4 scenes were generated with the following configuration:

- 3 min vertices
- 8 max vertices
- 0.05 min radius
- 0.3 max radius

The setting that they differed in was the number of polygons. Scene 1 has 5, Scene 2 has 8, Scene 3 has 16, and Scene 4 has 32.

Approach

The approach followed the strategy mentioned in the assignment. The program generates a convex polygon by first determining the number of vertices within a given range. Then, it generates random angles (in radians) and radii for each vertex. Using these angles and radii, it calculates the x and y coordinates of each vertex and stores them in the vertices array. The ConvexHull function from scipy.spatial is used to compute the convex hull of the generated vertices, ensuring the resulting polygon is convex.

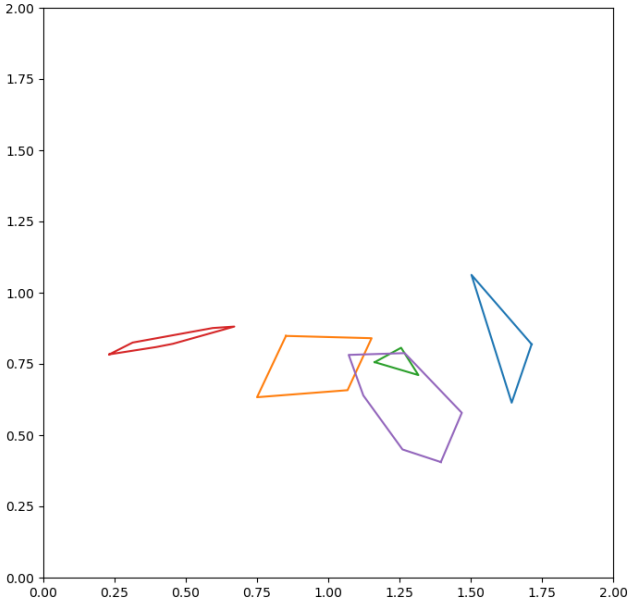


Figure 1: Scene 1

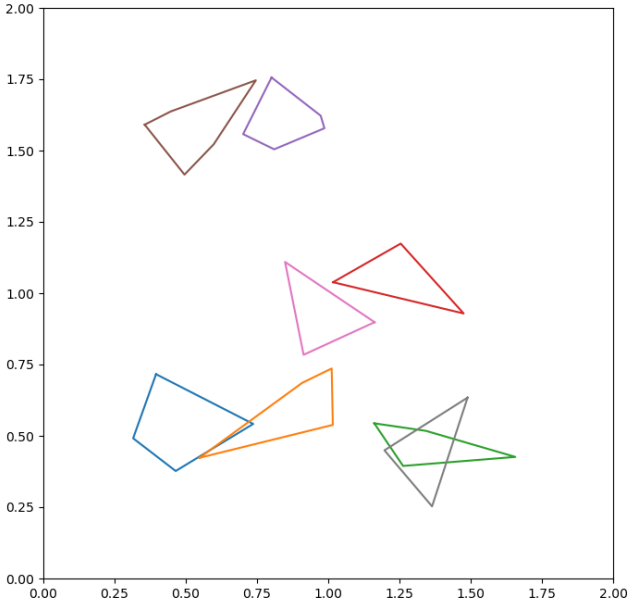


Figure 2: Scene 2

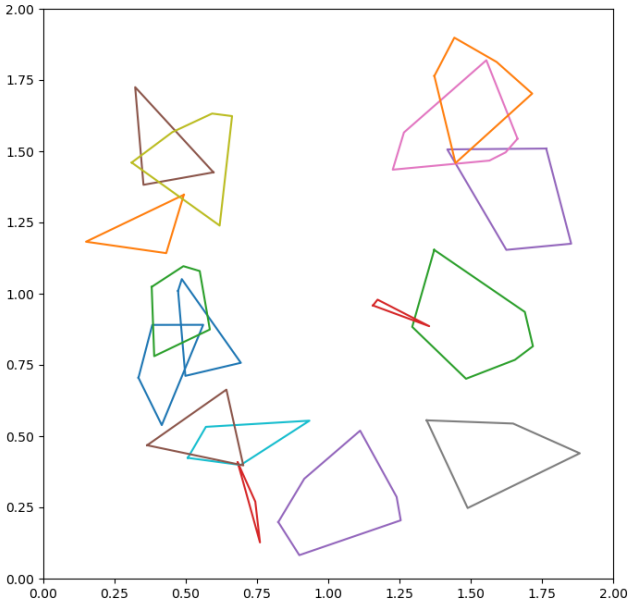


Figure 3: Scene 3

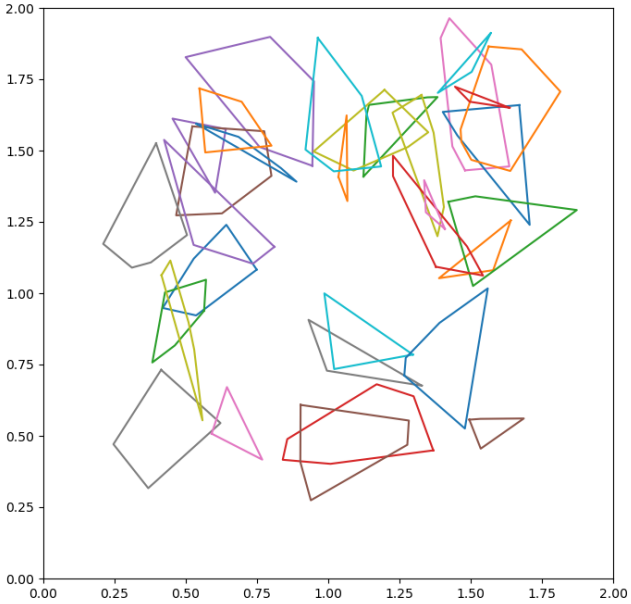


Figure 4: Scene 4

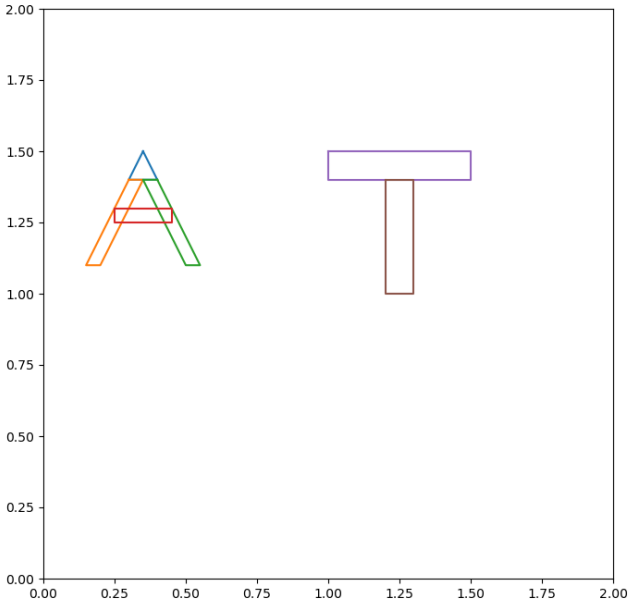


Figure 5: Scene 5

2 2D Collision Checking for Convex Polygons

Approach

The approach followed the mentioned strategy in the assignment. The program first checks all pairs of line segments between polygons for collisions. It then checks if any of the vertices are inside another polygon using a ray casting algorithm. This handles the case where a polygon is completely inside another one. Any polygons that are not marked as intersecting after these checks are plotted as unfilled. Otherwise, they are plotted as filled. Below are the collision plots of the 5 figures from Part 1:

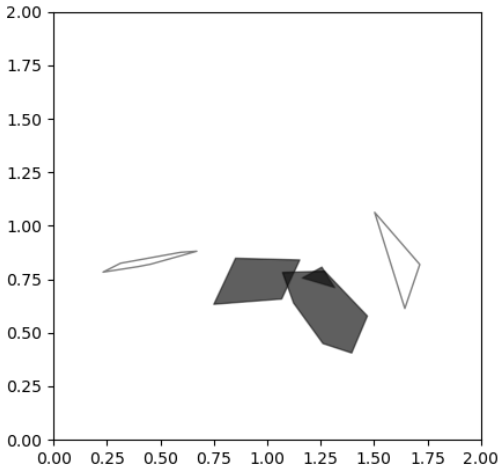


Figure 6: Collision Plot 1

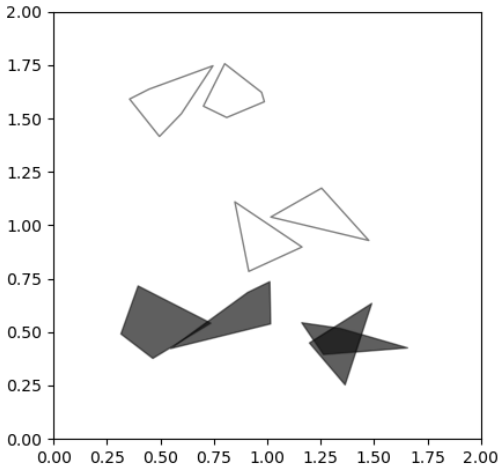


Figure 7: Collision Plot 2

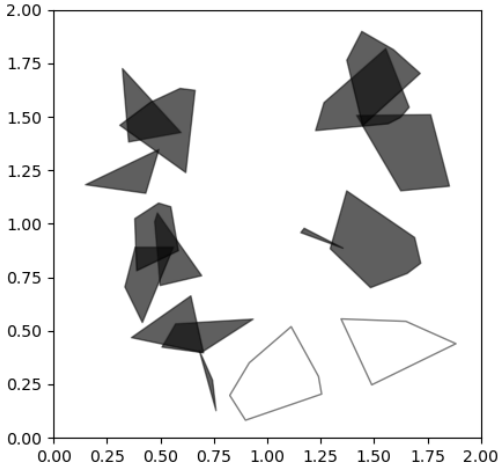


Figure 8: Collision Plot 3

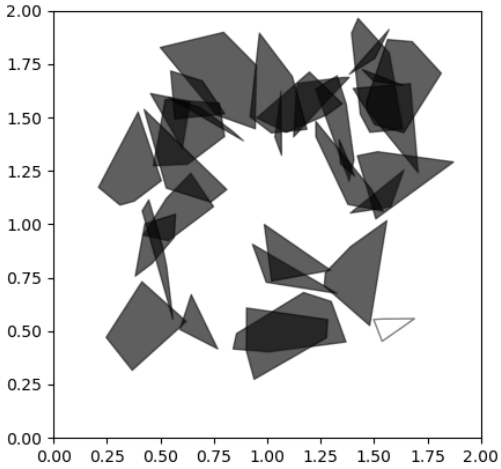


Figure 9: Collision Plot 4

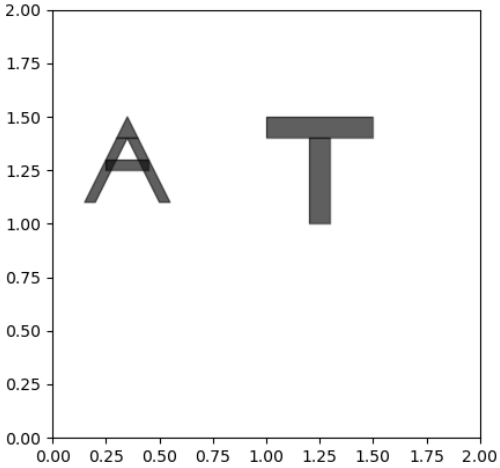


Figure 10: Collision Plot 5

3 Collision-Free Navigation for a 2D Rigid Body

The four keys used to control the rectangle on its own axes are:

- 'w' for forward
- 'z' for backward
- 'a' for anticlockwise
- 'd' for clockwise

Scenes with Different Orientations

Computing the Minkowski Sum in Scene Navigation

Minkowski Sum: A Brief Overview

The Minkowski sum is a mathematical operation that combines two shapes, P and Q, to create a new shape, $P \quad Q$, that represents all possible translations of Q that maintains contact with P without penetrating it. When considering convex polygons, such as rectangles, and another polygon (either convex or non-convex), this operation can be described geometrically and algebraically.

Geometric Explanation

Given two sets of points $P = fp_1; p_2; : : : ; p_n g$ and $Q = fq_1; q_2; : : : ; q_m g$, the Minkowski sum, denoted as $P \quad Q$, is computed as:

$$P \quad Q = fp + q \, j \, p \; 2 \; P; q \; 2 \; Qg$$

This means that the Minkowski sum is formed by adding each point q in Q to each point p in P, resulting in a new set of points that defines a new shape.

Algorithmic Explanation

In the context of robot navigation within a scene containing obstacles:

1. For each obstacle polygon in the scene, extract its vertices P.
2. Define a reference polygon Q (e.g., the robot shape, often a rectangle), extract its vertices, and possibly rotate it according to the desired orientation.
3. Compute the Minkowski sum $P \quad Q$ by adding every vertex $q \; 2 \; Q$ to every vertex $p \; 2 \; P$. The result is a new set of vertices that define a new polygon.
4. (Optional) To obtain a clean, non-intersecting polygon from the Minkowski sum, compute the convex hull of the resulting set of points.
5. The resulting polygon represents the configuration space obstacle, a region that the reference point of Q must avoid to prevent collision with P.

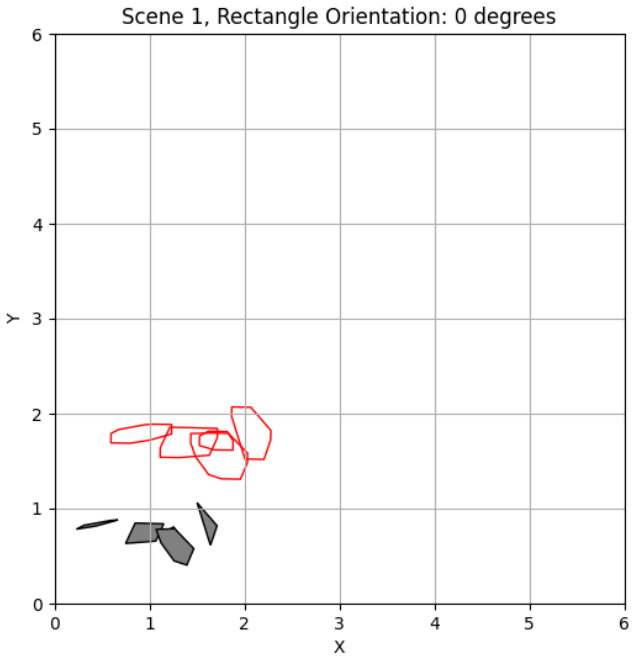


Figure 11: Scene 1, Orientation 1

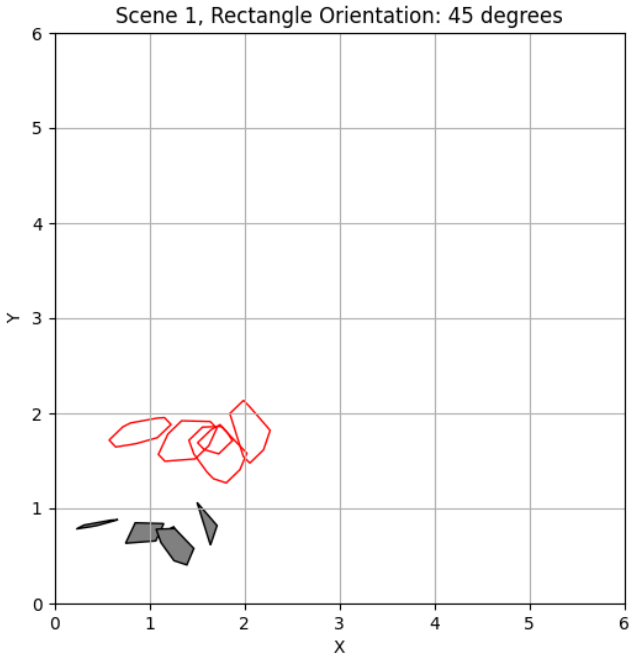


Figure 12: Scene 1, Orientation 2

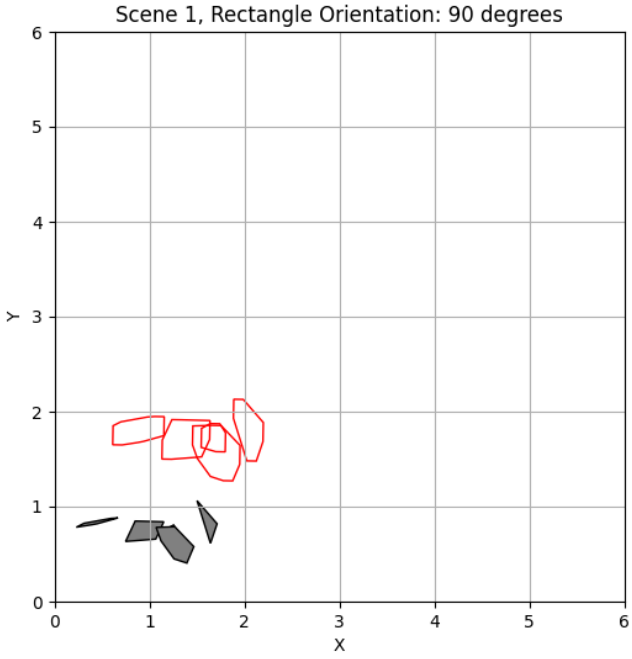


Figure 13: Scene 1, Orientation 3

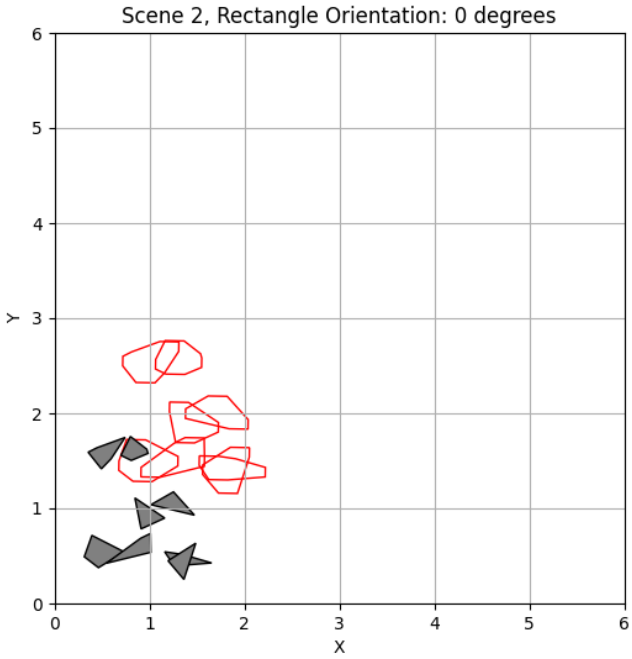


Figure 14: Scene 2, Orientation 1

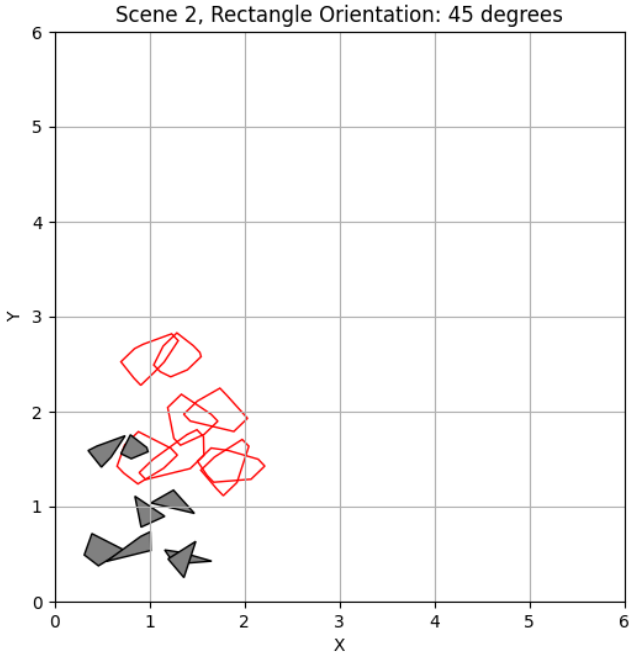


Figure 15: Scene 2, Orientation 2

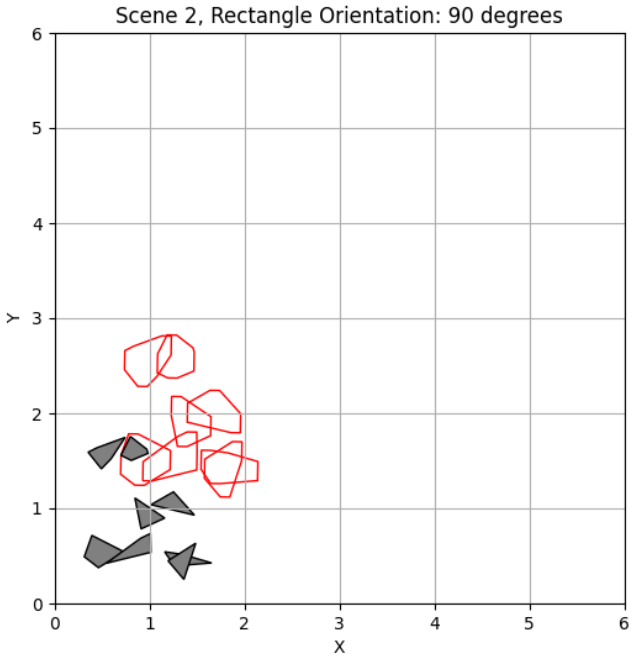


Figure 16: Scene 2, Orientation 3

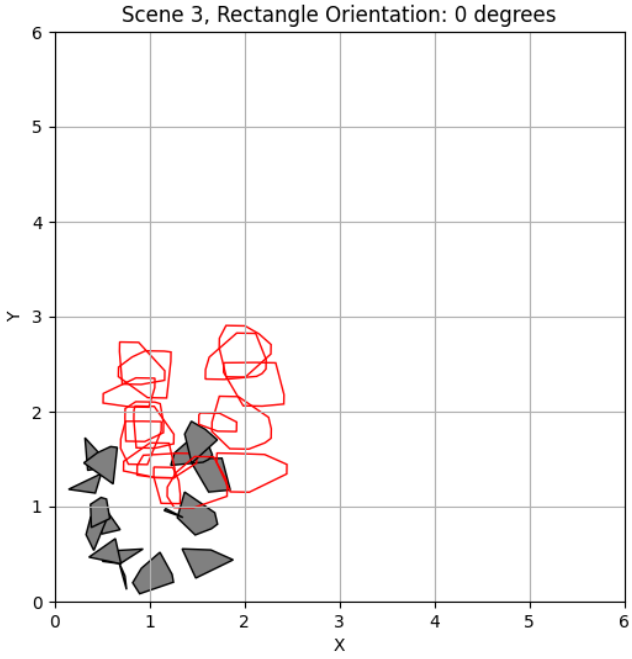


Figure 17: Scene 3, Orientation 1

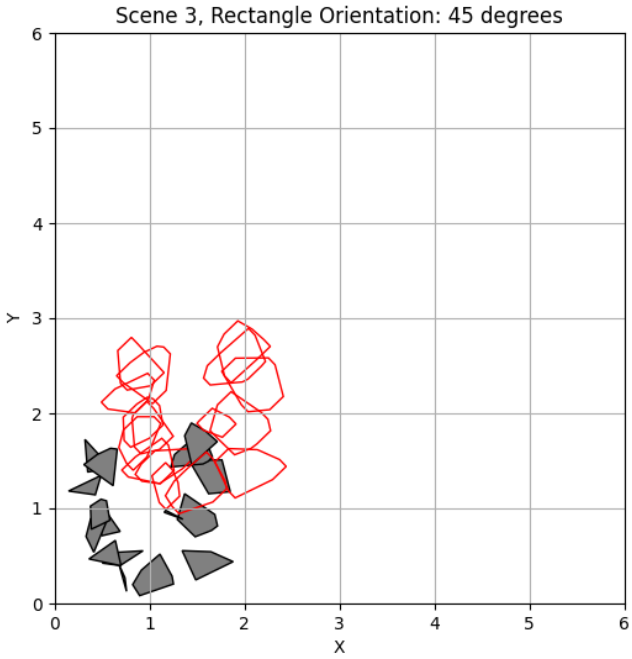


Figure 18: Scene 3, Orientation 2

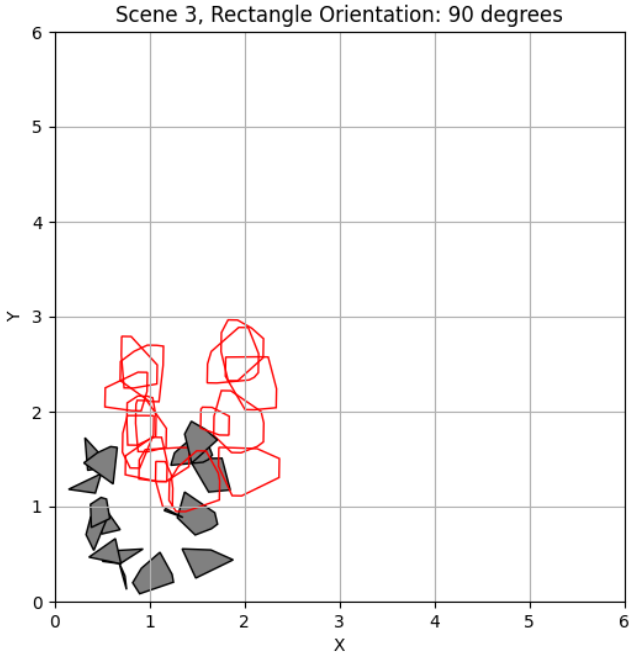


Figure 19: Scene 3, Orientation 3

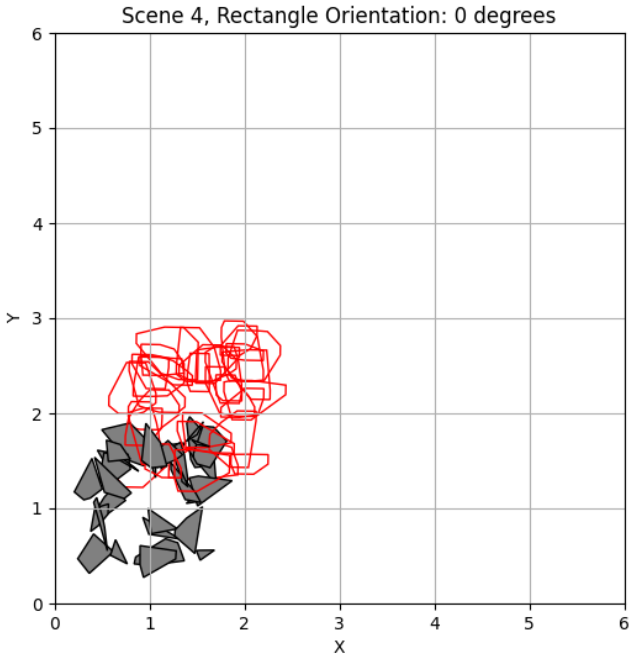


Figure 20: Scene 4, Orientation 1

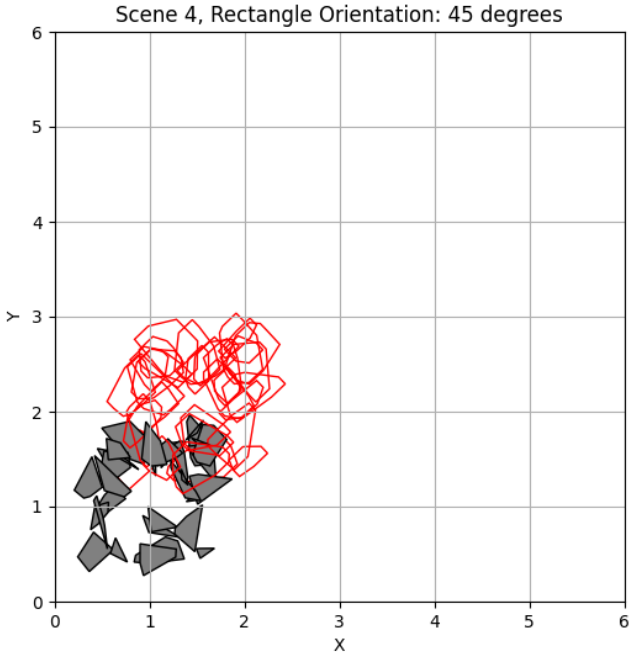


Figure 21: Scene 4, Orientation 2

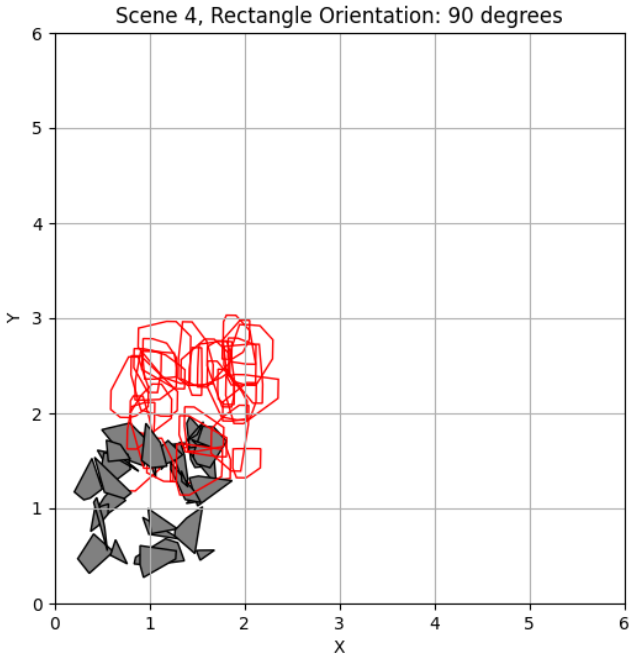


Figure 22: Scene 4, Orientation 3

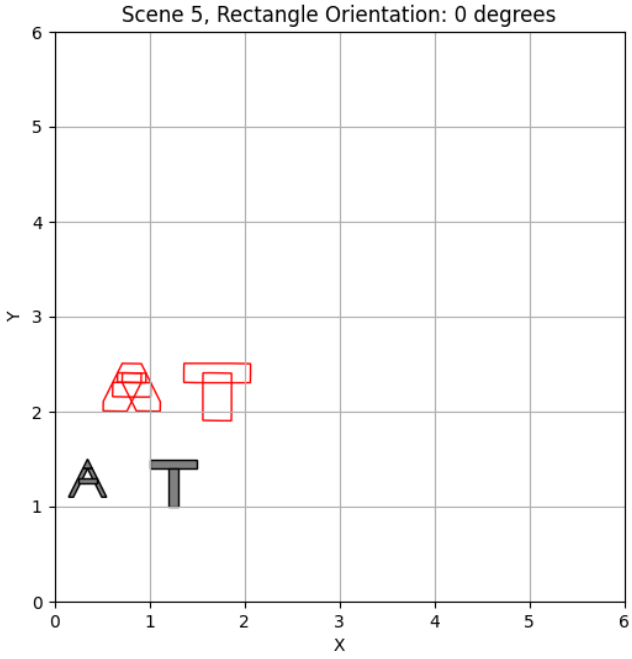


Figure 23: Scene 5, Orientation 1

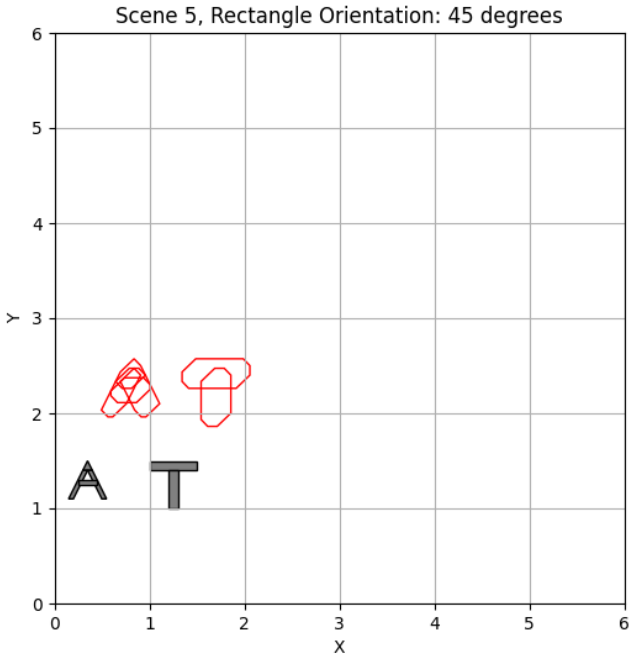


Figure 24: Scene 5, Orientation 2

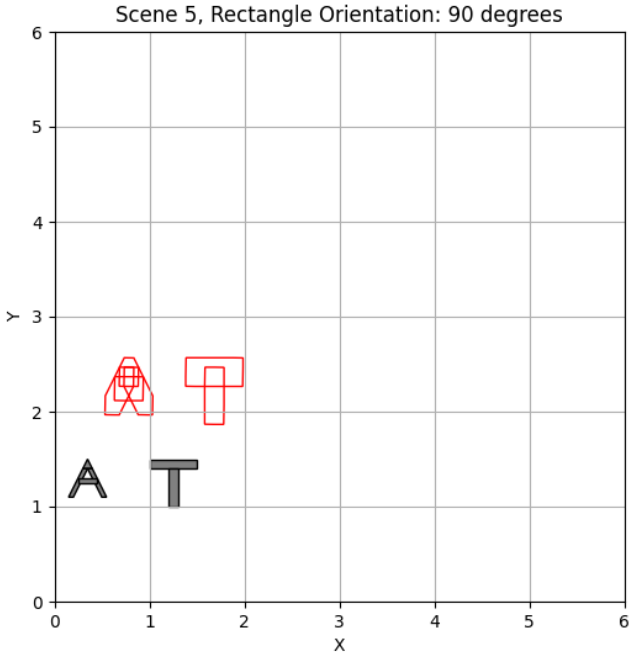
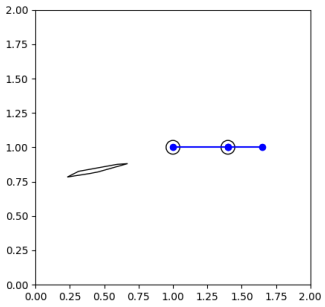


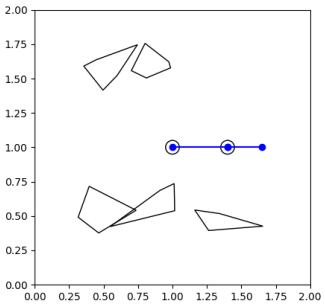
Figure 25: Scene 5, Orientation 3

4 Collision-free Movement of a Planar Arm

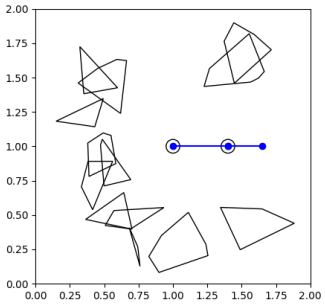
Initialized arms and obstacles:



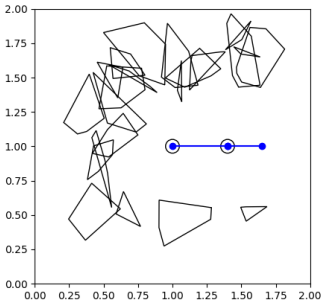
(a) Scene 1



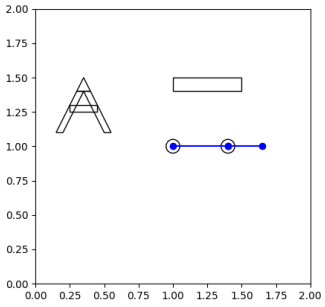
(b) Scene 2



(c) Scene 3



(d) Scene 4



(e) Scene 5

Figure 26: Polygonal scenes with a planar arm, showing different configurations.

Configuration spaces for planar arm: