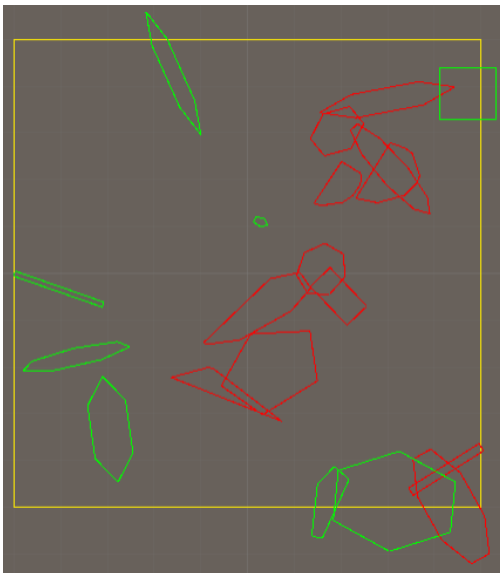


Unity Collisions

1. This is a group assignment.
2. Clone [this repository](#) from GitLab and set up your own **private** repository on GitHub with all members. If you are unfamiliar with Git, please refer to the attached tutorial. Each group project will be set up on a different repository, and your group will submit a log of your commits using the command below. **Your commit messages should be descriptive.**

```
git log --pretty=format:'%h was %an, %ar, message: %s' > log.txt
```

3. Open the scene called **Collision Scene**, and familiarize yourselves with this testbed for your collision assignment. Based on the parameters in the Prism Manager, regular and irregular convex polygons will be spawned in the scene (if Prism Region Radius = 0 and Max Prism Scale Y = 0). The goal of this assignment is to re-write certain functions in the code base to resolve all of the collisions in the scene.



4. [7 points] Re-write the function `PotentialCollisions()`. This function is responsible for picking out unique pairs of polygons that you want to check collisions for. The default implementation is brute-force, which is unacceptable. Your implementation must be $O(n \log n)$ w.r.t. time. If you **implement/borrow (specify which)** a data structure for nearest neighbor queries, e.g., k-d tree, bin-lattice spatial subdivision, quadtree, locality sensitive hashing, octree, etc., visualize the data structure by drawing debug lines. **Write a concise paragraph explaining the efficiency of your approach.**

5. [7 points] Re-write the function `CheckCollision()`. For each pair of shapes that is potentially colliding, this function must first determine whether there is an actual collision using the GJK algorithm (**implement from scratch**). If there is an actual collision, compute the penetration depth vector from the first shape to the second shape using the EPA algorithm (**implement from scratch**).
6. Submit the following in Sakai for grading:
- Your Unity project in a zip file which contains the "Assets/" folder and includes all of your C# scripts.
 - A **link** to a video demo of your game with PrismManager parameters: 50, 5, 0, 3, 0.
 - A document (text or pdf) containing:
 - The text response to (4).
 - A description of the extra credit attempts.
7. Extra credit opportunities:
- [5 points] Implement one of the data structures named in (4) from scratch and integrate it.
 - [2.5 points] Borrow an existing implementation and integrate it. (**You must cite where you found the implementation.**)
 - [5 points] Set `Prism Region Radius = 5` and `Max Prism Scale Y = 3`, and complete (4) in 3D. For this, collisions can be resolved on the XZ plane.
 - [10 points] Set `Prism Region Radius = 5` and `Max Prism Scale Y = 3`, complete (4) in 3D, and complete (5) in using 3D GJK and 3D EPA.

