```csharp
private Vector2[] GetRegionBorders(RegionData region)
{
    // Grab every hex in the region and take the first
    var hexes = _regionHexes[region.Id];
    var initialHex = hexes.First();

    // initialise a new list to represent hexes we've seen and add our
    // first hex to it. we only want to merge each hex in once
    var seen = new List<Vector2I> { initialHex };

    // grab the polygon points with another function
    var polygonPoints = GetHexLocalPolygonPoints(initialHex);

    // recursively merge those points with the neighbours
    // and return the result
    return MergeNeighbourPolygons(
        polygonPoints,
        initialHex,
        seen,
        hexes);
}

private Vector2[] GetHexLocalPolygonPoints(Vector2I hex)
{
    // to find our collision polygon, we grab the cell data and use that
    // to get the collision polygon points in list format
    var cell = _baseLayer.GetCellTileData(hex);

    // the polygon points are a list of offsets from the centre
    var cellPolygon = cell
        .GetCollisionPolygonPoints(0, 0)
        .ToList();

    // we get the local (screen) coordinates(e.g. (50px, 25px)) of the
    // hex rather than just its tile coordinates (e.g. (12, 0))
    var local = _baseLayer.MapToLocal(hex);

    // add the local coords to each polygon offset
    for (var i = 0; i < cellPolygon.Count; i++)
    {
        cellPolygon[i] = cellPolygon[i] + local;
    }

    return cellPolygon.ToArray();
}

private Vector2[] MergeNeighbourPolygons(
    Vector2[] initialPolygon,
    Vector2I initialCell,
    List<Vector2I> seen,
    List<Vector2I> regionCells)
{
    var resultPolygon = initialPolygon;

    // we check each neighbour of our initial hex
    // that fulfills these criteria:
    // 1. is in the same region
    // 2. has not been seen before
    foreach (var neighbour in _baseLayer
        .GetSurroundingCells(initialCell)
        .Where(c =>
            regionCells.Contains(c) && !seen.Contains(c)))
    {
        // add that neighbour to seen
        seen.Add(neighbour);

    // get the polygon points for the neighbour
        var neighbourPolygon =
            GetHexLocalPolygonPoints(neighbour);

    // merge it into our result polygon using Godot's
    // Geometry2D.MergePolygons
    var merge =
        Geometry2D.MergePolygons(
            resultPolygon,
            neighbourPolygon.ToArray());

    // if the merge was unsuccessful, this array would be more than 1 item
    // but I've set it up so it can always merge by only using neighbours
    resultPolygon = merge[0];

    // now recursively call this function with the neighbour as the
    // initial hex, so it gets all the neighbour's neighbours
    resultPolygon =
        MergeNeighbourPolygons(
            resultPolygon,
            neighbour,
            seen,
            regionCells);
}

    // once everything's been seen, we'll be left with our single polygon
    return resultPolygon;
}
```