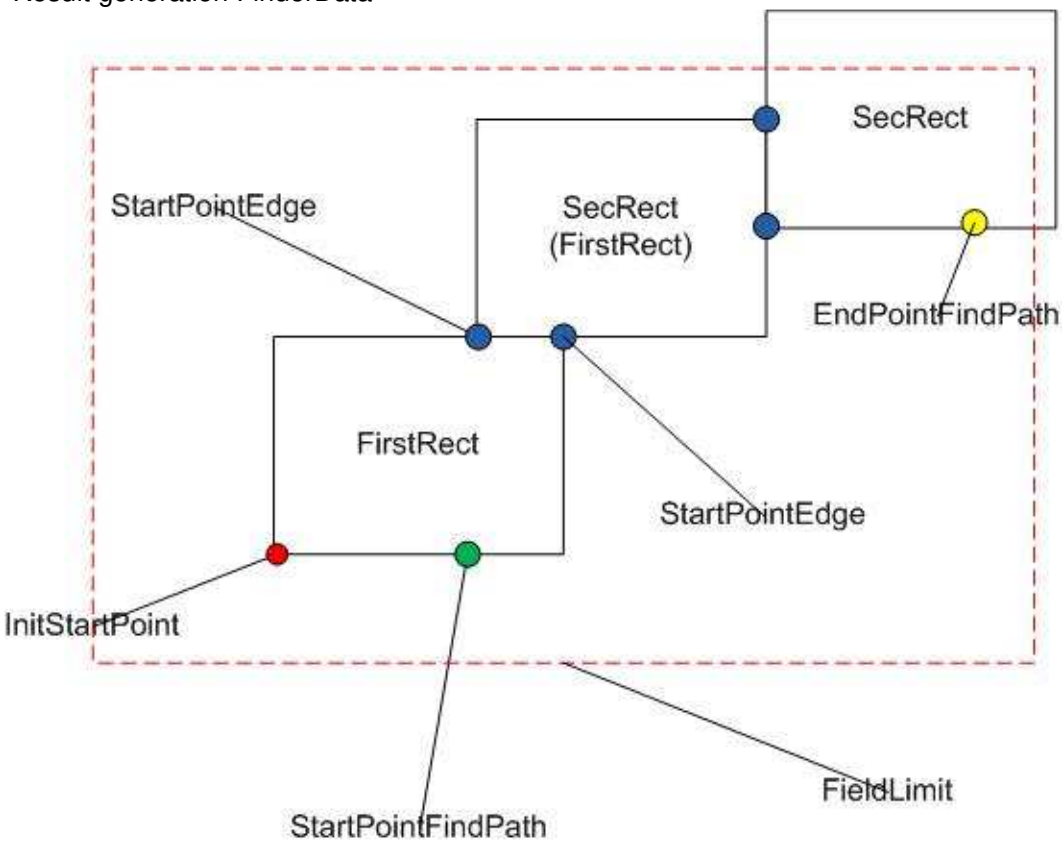
Service for finding a path through the given edges of intersections of rectangles with the least number of turns.

PathFinder include two functionality:
- Generation the FinderData – generation of random edges of intersections of rectangles
- Find Path - finding a path through the edges

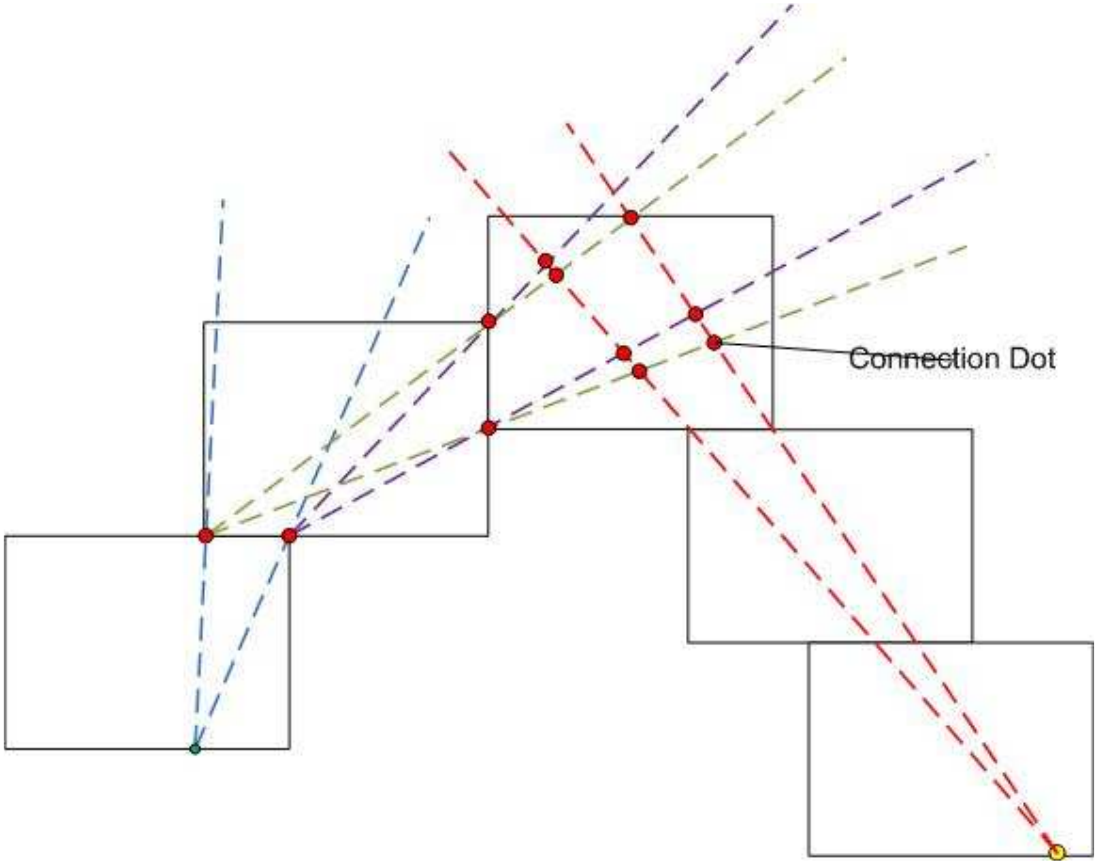
Used the next structure of FinderData and call to get the path:

```
struct Rectangle
{
	public Vector2 Min;
	public Vector2 Max;
}
struct Edge
{
	public Rectangle First;
	public Rectangle Second;
	public Vector2 Start;
	public Vector2 End;
}
interface IPathFinder
{
	IEnumerable<Vector2> GetPath(Vector2 StartPath, Vector2 EndPath, IEnumerable<Edge>
	edges);
}
```

Result generation FinderData



Result get the path – connection Dots from StartPath till EndPath.



Note. Algorithm find the collection of Connection dots (in Graph format), with minimum turns, but didn't make any Graph optimization (like find the shortest path etc.). Simply will get the first possible connection dot begin from last point (endPath). Oder to put connection dots not predeterminate.

The main concept:

From initial dots created the SectorSolutions (from Start and End Path points) based on closest Edge accessible from these dots. The SectorSolution limited by lines based on baseDots and passed through edge point («all solution can exist only into these Sectors»)

After that trying to find different intersection between these solutions ("a line is the best solution in always"), therefore the line connected two solutions is the solution.

If the current position of SectorSolutions (SolutionForDots for Start and End Path points) we can't find the connection line, we move SectorSolutions (SolutionForDots for Start points) to more farthest edges accessible from current its position (edge). And After repeat process to find different intersection between these solutions.

When we Create SectorSolution (initially or move solution for start point) we always create connectionsDots which connect these solution with Initial Dots and with previous corresponding solutions.

When we find the intersection between solution the create correspondent connection Dots.

All connectionsDots have information about previous ConnectionDot (or Dots) and all begin from ConnectionDot for Start PathPoint, The Last ConnectionDot contains the End PathPoint (It's a Graph).

Moving through these Graph from end to start we can find many ways which connected the Start and End Path Points. Also, it gives a possibility (if it will demands) to find optimal way by use this Graph.

The main objects and options to run:
Project have one Scene in Assets/Game/_Scenes: **PathFinder.unity**

The main start script (GameObject PathFinderTester) PathFinderTester : MonoBehaviour:
- Call the Generation (GeneratePathFinderData.GenerateData())
The result will be stored in class PathFinderData
- and after Call method Finder.CheckDataAndGetPath(), to check the generated data and to find path
Note. Because the generation fully randomized - some generated data can't be used to get path (in 50% cases the rectangles which form edges overlapping each with other)

GameObject PathFinderTester:
- have button in Inspector to initiate the regeneration of data and restart new find path on new data.
- **have different options**:
-- Use Seed from Field Setting (the generation will be not randomized, it will use some seed from GenerationSettingSO, see below).
-- Show Graph Path (after found path) to console will be outputted the final Graph of Connection Dots
-- TurnOnDebugPathFinder – turn on full tracing of process of finding the path (include output to console and show on Scene the "main graphical temporary lines and dots", which was used to find path.
(Also demanded to set the **key DEBUGFINDER** in Player Setting of the project). Will automatically create the GameObjects "DebugPathFinder", which will parent for all debug objects ("main graphical temporary lines and dots")
Also, in GameObject PathFinderTester (Show Path componet), options:
- DrawStepsPath show on Scene the lines (result of founded path) which connect Start with End Path point
- DebugLogPointsPath output to console the connection dots of founded path, which use to draw lines ("DrawStepsPath")
Note. Also, will show in readonly mode the object PathFounded (result)

GameObject [GenerateFinderData] will parent for all objects will build in process of generation the FinderData (rectangles and any dots), also **have different options**:
- contains the used GenerationSettingSO (see below)
- Use Seed from Field Setting (value can be override by value from **GameObject PathFinderTester**)
- ShowUsedRandomSeed – will output to console the current Seed which will use to current Generation of FinderData.
Note. Also, will show in readonly mode the object FinderData (after generation)

GenerationSettingSO contain setting which determinate the generation of FinderData, main options:
- MinNumberEdges – the generator will adjust the min size (width/height) Rectangle to guarantee the possibility (in any randomized values) the put demanded number of rectangles on current Field (limited by width/height)

- NotOutFromFieldSize – in case of possibility (specific randomized values) to put the maximum possible number of rectangles on current Field
- MaxNumberEdges – limit the maximum number of edges (rectangles-1) on current Field
- Min/Max percentage edge - will constrain the portion of the edge used to find the path from the actual size of the edges of the rectangles that form that edge.

Note. Demanded to set the **key GENERATEDEBUG** in Player Setting of the project), if you that to receive debug output in process of Generation FinderData.

Main Classes

Exist two types of Solutions:
SolutionForDot : ISolution
SolutionForEdgeForStartPoint : ISolution

Each Solution have:
private readonly SectorSolutions _sectorSolutions; (or SectorSolutions[] in case "SolutionForEdge")
private readonly int _numLastCrossedEdge;
private readonly int _numRecBaseDot;
private readonly ConnectionDot _connectionDot; (or ConnectionDot[] in case "SolutionForEdge")

Each SectorSolutions have: (its s sector of possible solutions limited by two line, both of which start from the point baseDotSectorSolutions)
public readonly Line LineB;
public readonly Line LineA;
public readonly Vector2 baseDotSectorSolutions;

Each Line have (factorX * X + factorY * Y = factorB):
protected readonly float _factorX;
protected readonly float _factorY;
protected readonly float _factorB;

but most comonly used the NormolizedLine where FactorY = 1f; (factorX * X + 1 * Y = factorB)

Exist special Type of Line:
LineVertical : Line
private const float FactorXVerticalLine = 1f;
private const float FactorYVerticalLine = 0;

LineHorizontal : Line
private const float FactorXVerticalLine = 0;
private const float FactorYVerticalLine = 1f;

Each Solution have a ConnectionDot (or ConnectionDot[]) Store the connection between different Solutions (connection between "baseDotSectorSolutions")
public readonly Vector2 baseDot;
public readonly IEnumerable<ConnectionDot> prevConnectionDots; //can connected to more than one of other ConnectionDot
Note. For simplify the ConnectionDot.baseDot separated from SectorSolutions.baseDotSectorSolutions (but in most cases it is one point, but not in all cases)

ListDotsPath

```
private static List<ConnectionDot> _list;
private static List<Vector2> _path;
```

Short Description of the Process Path finding

Initialization for Cycle:

- static class StoreInfoEdges.InitStoreInfoEdges()
- static class ListDotsPath.InitListDotsPath()
- create initial SolutionForDot for:_startPointFindPath &_endPointFindPath

Cycle:

- Any method trying to find path which can connect current solution for StartPoint with solution for endPoint

- All methods return true if they found path (in other case false), Order of methods is important:

```
TryLinkCurrentBaseDotSolutionStartWithEndPoint();
IsBothSolutionOnOneEdge();
TryCrossingCurrentSolutionWithSolutionForEndPoint();
TryCreateDirectLineLinkedDotsCurrentSolutionWithSolutionForEndPoint();
```

- If all methods did not find the path (all returned false) the next iteration will be move the "Solution for StartPoint" to the endPoint more close

```
SolutionForEdgeForStartPoint.CreateNewSolutionForEdge(_currentSolutionForStartPoint,
_arredges.Length - 1);
```

- In case of error in found the Path (in case of correct initial Data it's not possible) in method SolutionForEdgeForStartPoint.CreateNewSolutionForEdge, exist check:

```
if (numEdgeCurrentSolution == farthestNumEdge)
       throw new NotSupportedException("Something wrong, because in this case the
Finder.IsBothSolutionOnOneEdge() should have been called before");
```

It means that the intial Data contain the problem which we not detected at checking it