Documentation

# Waypoint.cs

Waypoint must exist within a loop to operate waypoint is the basic component of the system it connects to other waypoint through loop or branches

## Variables:

private int \_id :unique id based on the position"

public WaypointLoop parent :the loop that contains the waypoint"

string json: stores the data of the waypoint in a json format"

public Waypoint Next: the next waypoint in the path, null if the last waypoint"

public Waypoint previous: the previous waypoint in the path, null if the first waypoint"

public Transform HandleA: the handle that control the previous link\nused in creating bezier bath"

public Transform HandleB: the handle that control the next link\nused in creating bezier bath"

public bool LockHandles : True: handles work together as one line result in continous path at the point\nfalse: handles work separately result in a break in the continouity of the path"

private bool DrawLink: controls if the link to the next waypoint is visible or not"

private float localYoffset: temporary saved data for 2D loop"

private float HandleAoffset: temporary saved data for 2D loop"

private float HandleBoffset: temporary saved data for 2D loop"

public List<Bezier.PathPoint> inBetweenPoints: gets the intermediate path points between the the point and the next waypoint"

public Bezier.PathPoint[][] InBetweenBranches: gets the intermediate path points between the the point and the Branches"

public List<Waypoint> Branches: list of waypoints to connect to"

public List<Waypoint> ReverseBranch: list of waypoints that is connected to this point"

public bool enterance, exit: specify if the waypoint is an entrance of a branch of an exit

public Vector3 normalDir: vector directed to the normal to the waypoint, normat to the direction to the nect point"

public float distanceBetweenPoints = 1f :distane between intermediate path points between the point and the next point"

public event EventHandler onStateChanged: event is run when point change position or change direction or handles

this event is run whenever

public event EventHandler onDeleted: event is run when point is deleted"

public bool drawinbetween = true: draw in between points and normal

True: draw gizmos

False: Don’t Draw

public int GizmoMode: change gizmo mode

0: no Gizmo

1: only points

2: only lines

3: with normal

## Methods:

getInbetween:

### public List<Bezier.PathPoint> getInbetween(Waypoint Next)

return the inbetween point using the next waypoint

<param name="Next"></param>

<returns>list of bezier points, null if the Next is not connected to the current point</returns>

The inbetween points are basically the intermediate points when moving from point A and point B

### public List<Bezier.PathPoint> getInbetween(int id)

return the inbetween points using index of the waypoint

<param name="id">the index of the branch, -1 if you want next</param>

<returns>list of bezier points, null if the Next is not connected to the current point</returns>

### public static List<Bezier.PathPoint> getInbetween(Waypoint A,Waypoint B)

return the bezier path points between any two waypoints regardless the system

<param name="A">start point</param>

<param name="B">end point</param>

<returns>return list of points</returns>

OnStateChanged:

### public void OnStateChanged()

recalculate point parameters if state changed.

you can call it whenever you want

it recalculate the normal and the inbetween points and the Branches.

### public void OnDeleted()

clean up after deleting waypoint

### public Vector3 getForwardVec()

return the vector to the next point tangent to the curve

graph related:

### public void RecalculateBranches(List<Waypoint> Branches)

calculate in between opoints between waypoint and its brances.

called from the waypoint when changed state

<param name="Branches">list of waypoints to calculate hte inbetween, could be any points</param>

### public void RecalculateInBranches()

calculate in between opoints between waypoint and its brances.

called from the waypoint when changed state

### public void RecalculateReverseBranches()

calculate in between opoints between waypoint and its brances.

called by the brance when it changes state to notify the source of branch (waypoint)

### public void RecalculateInBetween()

calculate in between opoints between waypoint and its brances.

called from the waypoint when changed state

### public void recalculateNormal()

recalculate the direction of the normal based on the direction of the next and the right

### public void SetPosition(Vector3 position)

change position of the waypoint

must choose this if you need to change positiotn from code

### public void AddBranch(Waypoint branch)

connect waypoint as a branch

### public void RemoveBranch(Waypoint branch)

removes branch from loop

### public void CreateHandles()

set up handles at waypoint creation

### public void UpdateHandle( bool updateA)

update the handle based on the other handle state

this function works on lockHandle mode to ensure the handles are in the same line

<param name="updateA">true if you want to update handleA, else update handleB</param>

# WaypointLoop.cs

## Variables:

### public List<Waypoint> waypoints;

"waypoints that construct the loop"

### public WaypointSystem parent;

the waypoint system that contains the loop

### public bool isClosedLoop;

link the last point to the start point

### private bool is2d;

convert between 2d flat loop or 3d loop

loop has all points in the same plane can be rotated by rotating the loop

### public List<Waypoint> entrances;

mark the branch end waypoints as an entrance to the loop

### public List<Waypoint> exits;

mark the branch source waypoints as an exit to the loop

## getters and setters:

### public bool IsClosedLoop

link the last point to the start point

### public bool Is2d

switch between 2d and 3d loops

## methods:

### public void SaveWaypointsYOffset()

save offsets of the flat plane position

### public void ScanLoop()

create a loop waypoints from its children waypoints

/\*children must have **Waypoint** component

### public void ClearWaypoints()

remove all waypoints in the loop

### public Waypoint AddWaypointAttEnd(Vector3 position)

adding new waypoint as a last point in loop connect it to the previous

<param name="position">position of the waypoint</param>

<returns>the created waypoint</returns>

### public List<Vector3> GetPathPointsFromPoint(int Beginindex, int LastIndex)

get all point by sequence from start point to end

<param name="Beginindex">start point index in loop</param>

<param name="LastIndex">positive if index from begining

,and negative if you want the index from last one</param>

<returns>list of positions from start position to end\nif begin is larger than end return empty list, if gegin equal the end and closed loop return all points else return null</returns>

### public List<Waypoint> GetPathWayPoints(int Beginindex, int LastIndex)

gets the waypoint list for path from point at Beginindex to point at LastIndex

<param name="LastIndex">positive if index from begining

,and negative if you want the index from last one</param>

<returns>list of positions from start position to end\nif begin is larger than end return empty list, if gegin equal the end and closed loop return all points else return null</returns>

### public void toggleLoop()

create new handles, new positions for waypoints, update new connections from last point to first

### public void RepositionLoopOrigin()

make loop postion in the center of the waypoints

### public void automaticSetup()

automatic setup of the loop points , set up handles, normals, branches, inbetween points

### public void updateLoopPoints()

update loop waypoints, position and normals.

used whenever you change anything regarding the loop

### public void RemovePoint(int index)

removes waypoint from loop

### public Waypoint GetClosestWaypoint(Vector3 point,float minsnap=1f)

used when you don't know witch waypoint is near the object

<param name="point">the position of the object</param>

<param name="minsnap">the max distance between the point and the waypoint to connect</param>

<returns>the nearst waypoint, null if there is no waypoints in range</returns>

### public void RemovePoint(Waypoint point)

remove waypoint from loop

### public void AddWaypointAtIndex(Vector3 position, int index)

add new waypointat position in specific index in loop

### public void AddWaypointAfter(Vector3 position, Waypoint point)

add waypoint after another waypoint in loop

### public Waypoint AddWaypoint(Vector3 position , int index)

add new waypointat position in specific index in loop

### public void Setup(WaypointSystem par)

set up loop inside a waypointsystem

# WaypointSystem.cs

## Variables:

### public float nsapDistance = 2;

snap distance when select or branch

### public int WorldSize;

determine the width of the system to determine the id,don't set this at any cost.

### public bool autoset

set the new created waypoint with pre-calculated values for handles and norma

### public bool freeMoveHandles

move the handles using free move in 3d instead of 3 axis movement

### public bool GraphUpdated

true if the graph is updated. if graph is not updated the path finding won't work correctly

### public event EventHandler onSystemChanged

event run when system update

### public ConnectionType curveType

not used in action

### public List<WaypointLoop> loops;

list of loops to create the path

### public Dictionary<Waypoint, List<WaypointLink>> waypointgraph;

graph to perform the path finding algorithm with

## Methods:

### public void OnSystemchanged()

called when you change the system

### public WaypointLoop AddLoop(Vector3 position)

create new loop at position

<param name="position"></param>

<returns></returns>

### public void ScanLoops()

scan all loops that are children of the system

### public void markEnEx()

mark waypoint as entrance if its the end point of branch

and exit if its source of branch

### public List<Bezier.PathPoint> GetSegmentPoints(Bezier.BezierSegment seg, float spacing,float resolution)

get segment path points

a segment is a link between two waypoints

</summary>

<param name="seg">the segment</param>

<param name="spacing">the distance between the result points</param>

<param name="resolution"></param>

<returns></returns>

### public void CreateGraph()

scan the system and create a updated graph

### private List<Waypoint> GetpathBetweenPoints(Waypoint source, Waypoint dest)

get path from source to destination if the points belong in the same loop

</summary>

<param name="source">start</param>

<param name="dest">end</param>

<returns></returns>

### public List<Bezier.BezierSegment> GetPathpoints(Waypoint source, Waypoint dest , bool Bezier=true)

return path points in bezier form of straight path

</summary>

<param name="source"></param>

<param name="dest"></param>

<param name="Bezier">true: bezier points, false: straight points</param>

<returns>list of points of path</returns>

### public List<Bezier.BezierSegment> GetPathStraight(Waypoint source,Waypoint dest)

return the evaluated path with direct path between waypoints

### public List<Bezier.BezierSegment> TraverseLoopSegments(int LoopIndex, int startPoint,int LastPoint)

### public List<Bezier.BezierSegment> GetPathSegments(List<Waypoint> points)

convert list of waypoint to bezier segment list

</summary>

<param name="points">path points</param>

<returns>list of bezier segments</returns>

### public List<Bezier.BezierSegment> GetPathpointsBezier(Waypoint source, Waypoint dest)

return list of bezier segments between source and destination

</summary>

<param name="source"></param>

<param name="dest"></param>

<returns></returns>

### public List<Waypoint> EvalGraph(Waypoint source ,Waypoint dest)

evaluate a path between two points in the whole system

</summary>

<param name="source">starting point</param>

<param name="dest">ending point</param>

<returns>list of waypoints from start to finish</returns>

## graphV alueSaver:

list of waypoint links that generate the graph, this class is just a saver to save the data of the graph till next startup

contains a point and list of waypointlink that it links to

### public Waypoint point;

### public List<WaypointLink> links;

## WaypointLink:

class that contains the data of each link of waypoint

contains the next waypoint , the distance from this point to the next, and the bezier curve length

### public Waypoint next;

### public float absoluteDistance, CurveDistance;

## Node:

### A\* path finding Node

### public Node parent;

### public Waypoint point;

### public float f, g, h;

## Bezier.cs:

### public static Vector3 GetCurveture(Vector3 A, Vector3 B, Vector3 C, Vector3 D, float t)

get the curvature measure of the bezier curve

</summary>

<param name="A">point 1</param>

<param name="B">point 2</param>

<param name="C">point 3</param>

<param name="D">point 4</param>

<param name="t">progress along the curve</param>

<returns>returns a normal vector to the curve with length propotional to the curve</returns>

### public static Vector3 GetCurveRadius(Vector3 A, Vector3 B, Vector3 C, Vector3 D, float t)

get the radius of the curve at point t

</summary>

<param name="A">point 1</param>

<param name="B">point 2</param>

<param name="C">point 3</param>

<param name="D">point 4</param>

<param name="t">progress along the curve</param>

<returns>returns the radius of the curve at point t</returns>

### public static List<PathPoint> EvalPath(List<BezierSegment> bezierSegments, float spacing, float resolution = 1)

evaluate path to create bezier caurve point at that have specific spacing