Imarching is an interface class.

## Marching Cubes

- Surface : float
- Cube : float[8]
- WindingOrder: int[ ]

+ Marching(surface: float);
+ Generate(voxels: IList<float>, width: int, height: int, depth: int, verts: IList<Vector3>, indices: IList<int>);
+ March(x: float, y: float, z: float, cube: float[], vertList: IList<Vector3>, indexList: IList<int>);
+ GetOffset(v1: float, v2: float);

## Triangulation

- FlagIndex;
- Offset;
- EdgeFlags;
- idx;

+ MarchingCubes(surface: float);
+ March(x: float, y: float, z: float, cube: float[], vertList: IList<Vector3>, indexList: IList<int>);

## IMarching

Vector3 Vertices[];
Int Vertice Indices;
Float isolevel;

+ Generate(voxels: IList<float>, width: int, height: int, depth: int, verts: IList<Vector3>, indices: IList<int>);

Marching Cubes 0 — 1 Triangulation
Triangulation 0 — 1 IMarching
Marching Cubes 0 — 1 IMarching
Marching Cubes 1 — 0 Handler

## Handler

- Length : int
- Particles : Vec4 [ ]
- Groups : HashModel [ ]
- TestDraw : int [ ]

+ OnEnable();
+ OnDisable();
+ GetParticles();
+ GetIndices();
+ Update();
+ GetBounds();
+ OnFlexUpdate(_particleData: FlexContainer.ParticleData );
+ OnDrawGizmos();

Handler 1 — 0
Handler 0 — 1 NVIDIA Flex

## Hash System

- PointIndice : int[ ]
- Intervalx : int
- Intervaly : int
- Intervalz : int
- Particles : Vec4 [ ]
- Bounds : bounds
- Radius : float

+ GroupByCells():
+ SetData(indices:int[], particles: Vector4[], bounds: Bounds, radius: float, groups: ref HashModel[]):
+ checkS(vertice: int, particleIndex: int);
+ ifExist(vertex: int, point: int);
+ ifDoesNotExist(vertice: int, particleIndex: int);
+ findID(particle: Vector4, _length: float, _indice: int );

- Vec4 [ ] is a 3 dimensional vector with particle mass data.

- Bounds is a range that particles can reach in 3D space.

Hash System 1 — 0 Handler

İlişki tablosu: -Handler bir tane nFlexe sahip nFlex hiç handler'a sahip değil.
-Handler geri kalan tablolardan hepsine bir tane olmak üzere sahiptir ama o classlar handler'a sahip değildir.
-Bir handler'ın birden çok particle'ı olabilir ama bir particle'ın yanlızca bir tane handler'ı vardır.

Handler 0 — 1 Particle Finder
Handler 0 — 1 Surface Recognizer

## Particle Finder

-

+FindDimentionalIntervalNum(bounds: Bounds, _radius: float);
+ FindId(objectX: Vector3, bounds: Bounds);
+findBoundary(particleIndice: int, radius: float, Distance: int);

Particle Finder has no attribute because it is a class that calculates and return value. Hence, it does not require any attribute

## EdgeConnection

int [,]

EdgeConnection

## EdgeDirection

int[,]

EdgeDirection
CubeEdgeFlags
TriangleConnectionTable

## Surface Recognizer

- Particles: Vec4 [ ]
- Groups: Hashmodel [ ]
- Bounds : bounds
- ParticleNeighBounds : bounds

+ SetData(particles: Vector4[], bounds: Bounds, groups: ref HashSystem.HashModel[], radius: float):
+ FindDistance(vertex: Vector3, point: Vector3):
+FindKernel(s: float);
+ FindWeightedX(centerParticle: Vector3, neighbourParticles: Vector3[], searchRadius: float );
+ IsSurfaceParticle(centerParticle: Vector3, neighbourParticles: Vector3[]);
+ FindNeigbourParticles(neighbourCells: int[]);
+ FindBoundary();
+ GetParticleBound();
+ FindAreaCells(insideCell: Bounds);
+FindID(insideCell: Vector3 );

Surface Recognizer 0 — 1 Zhu&Bridson
Surface Recognizer 0 — 1 Situational Surface Calculator

## Situational Surface Calculator

- Particles: Vec4 [ ]
- Groups: VertexIndex [ ]
- Bounds : bounds
- ParticleNeighBounds : bounds

+ findDistance(vertex: Vector3, point: Vector3):
+ findConstant(radius: float):
+ findGradientWeight(particle: Vector3, neighbour: Vector3, radius: float ):
+ findNeigbourParticles(neighbourCells: int[]):
+ findKernel(s: float):
+ findWeights(vertex: Vector3, particles: Vector3[], weights: float[], length: float):
+ weightedPos(particles: Vector3[], weights: float[]);
+ zhuAndBridson(vertex: Vector3, weighted: Vector3, radius: float);

## Zhu&Bridson

- vertex: Vector3
- weighted: Vector3
- radius: float

+ ZhuandBridson(vertex: Vector3, weighted: Vector3, radius: float)