CULEBRA v 2.0 Beta

Creepy things For Grasshopper

complicitMatter

CULEBRA v 2.0

User's Guide to Culebra

Culebra is a live agent based C# library (Wrapper around Culebra Java library) and plugin for Grasshopper. It A collection of objects and behaviors for creating dynamic multi agent interactions. 2D|3D Multi Object Behavior library focused on hybrid system interactions with custom Visualization, Data, and performance features.

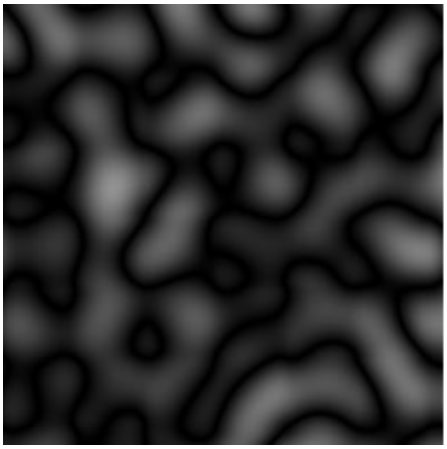
Culebra Concepts

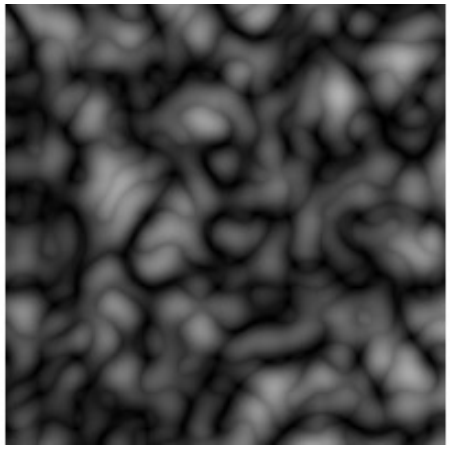
Multi Behavioral Systems

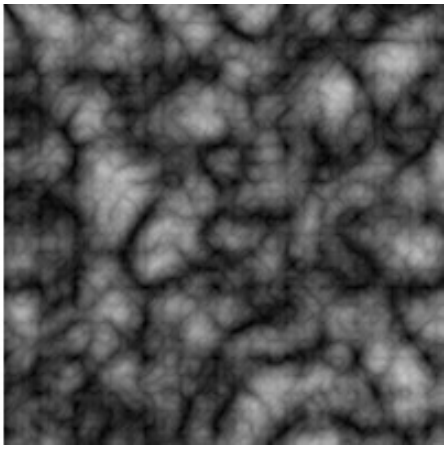
Perlin Noise

Perlin noise is a type of gradient noise developed by Ken Perlin in 1983 as a result of his frustration with the "machine-like" look of computer graphics at the time. To Ken Perlin for the development of Perlin Noise, a technique used to produce natural appearing textures on computer generated surfaces for motion picture visual effects. The development of Perlin Noise has allowed computer graphics artists to better represent the complexity of natural phenomena in visual effects for the motion picture industry.

Perlin noise is most commonly implemented as a two-, three- or four-dimensional function, but can be defined for any number of dimensions. An implementation typically involves three steps: grid definition with random gradient vectors, computation of the dot product between the distance-gradient vectors and interpolation between these values. - Wiki



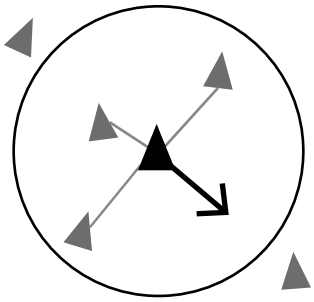




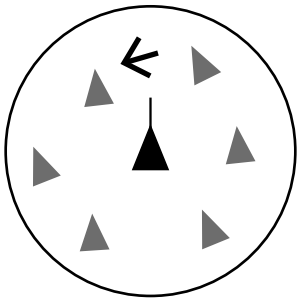
Flocking

Flocking behavior is the behavior exhibited when a group of birds, called a flock, are foraging or in flight. There are parallels with the shoaling behavior of fish, the swarming behavior of insects, and herd behavior of land animals. It is considered an emergent behavior arising from simple rules that are followed by individuals and does not involve any central coordination. - Wiki

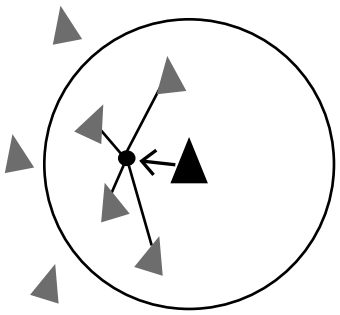
Basic models of flocking behavior are controlled by three simple rules:



Separation:
Avoid crowding neighbors (short range repulsion)



Alignment:
Steer towards average heading of neighbors



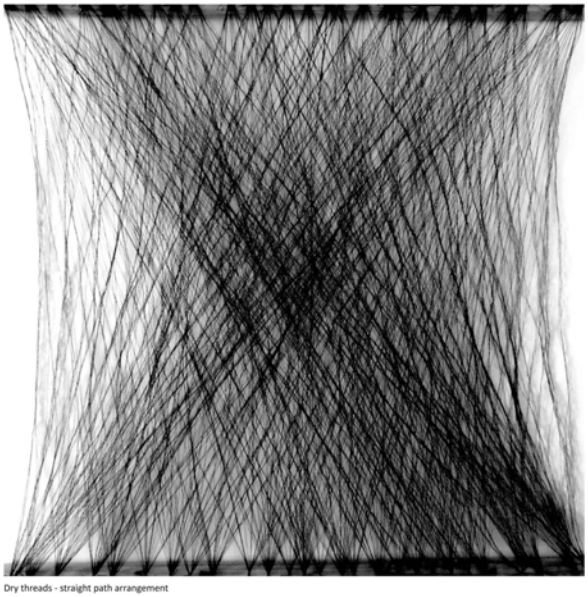
Cohesion:
Steer towards the average position of neighbors (long range attraction)

With these three simple rules, the flock moves in an extremely realistic way, creating complex motion and interaction that would be extremely hard to create otherwise.

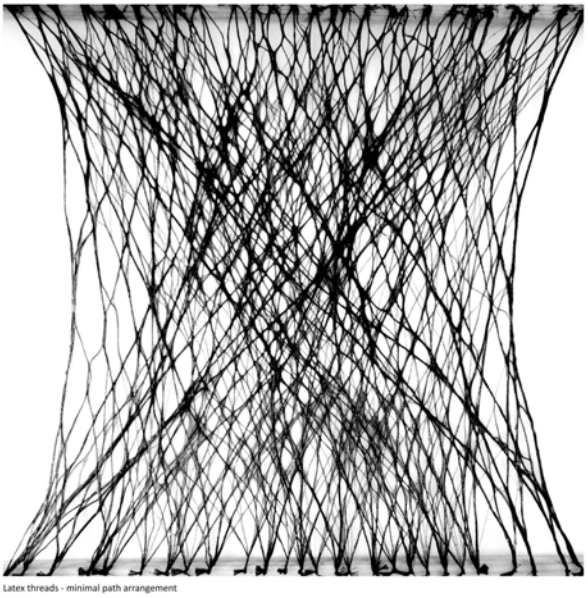
Self Organization

Self-organization is a process where some form of overall order or coordination arises out of the local interactions between smaller component parts of an initially disordered system. The process of self-organization can be spontaneous, and it is not necessarily controlled by any auxiliary agent outside of the system.

The original principle of self-organization was formulated in 1947 by the cybernetician William Ross Ashby. It states that any deterministic dynamic system will automatically evolve towards a state of equilibrium that can be described in terms of an attractor in a basin of surrounding states. Once there, the further evolution of the system is constrained to remain in the attractor. This constraint on the system as a whole implies a form of mutual dependency or coordination between its constituent components or "subsystems". - Wiki



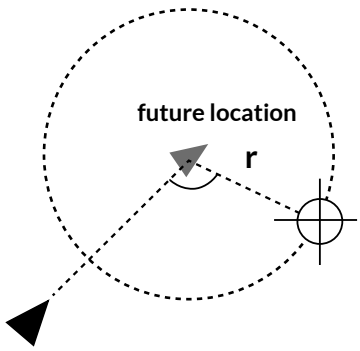
Dry threads - straight path arrangement



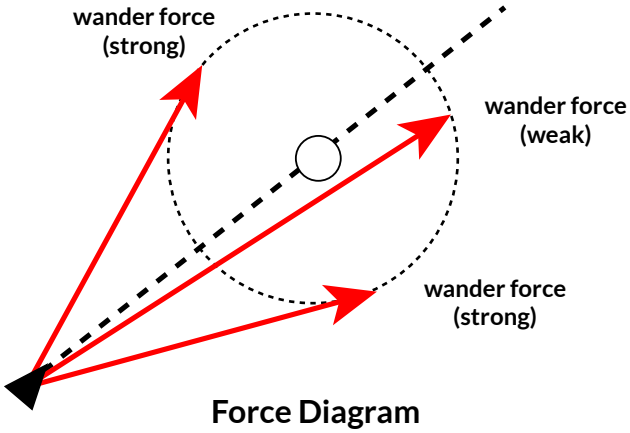
Latex threads - minimal path arrangement

Wandering

"Wandering is a type of random steering which has some long term order: the steering direction on one frame is related to the steering direction on the next frame. This produces more interesting motion than, for example, simply generating a random steering direction each frame."- Craig Reynolds



Future Location

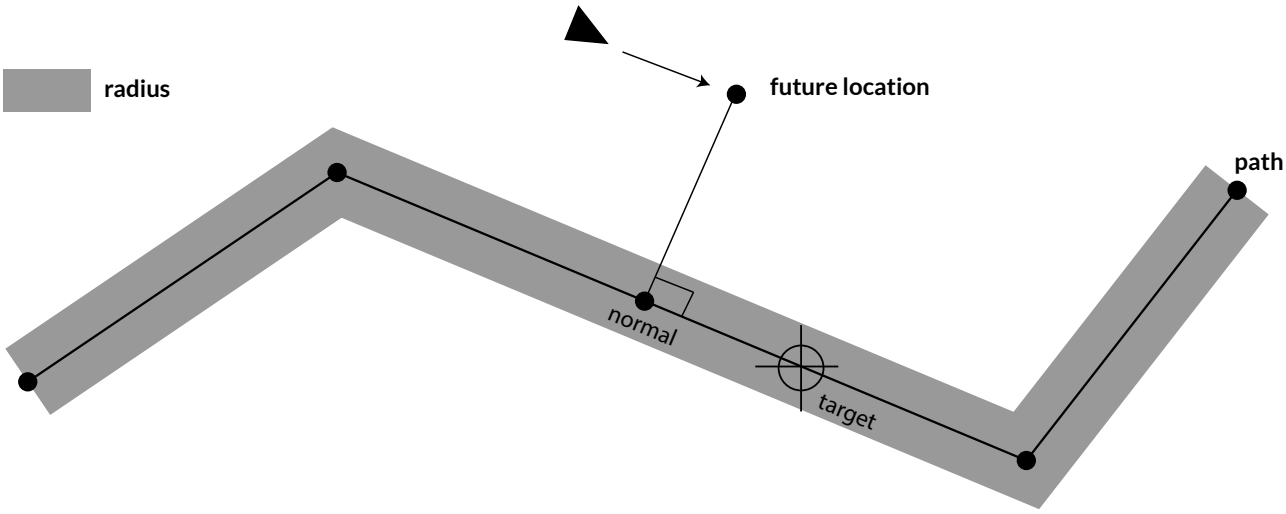


Force Diagram

"The vehicle predicts its future location as a fixed distance in front of it (in the direction of its velocity), draws a circle with radius r at that location, and picks a random point along the circumference of the circle. That random point moves randomly around the circle in each frame of animation. And that random point is the vehicle's target, its desired vector pointing in that direction" - Daniel Shiffman

Path Following

The task is to traverse the path in a given direction (entering on the left, exiting on the right) while keeping its center on the gray region. In this case the path is defined by a series of connected line segments (a polyline) and a radius. This behavior is related to containment and wall following but differs because the path implies a direction of travel. A non-zero path radius produces a sort of "sloppy path following". That is, the path following goal is considered to be met as long as the vehicle is within a certain neighborhood of the path spine. The path following requirement can be made more strict by reducing the radius toward zero. In general, the vehicle cannot be forced to always be exactly on the path if the path can change direction faster than the vehicle (as shown here, where the path makes sharp corners). - Craig Reynolds



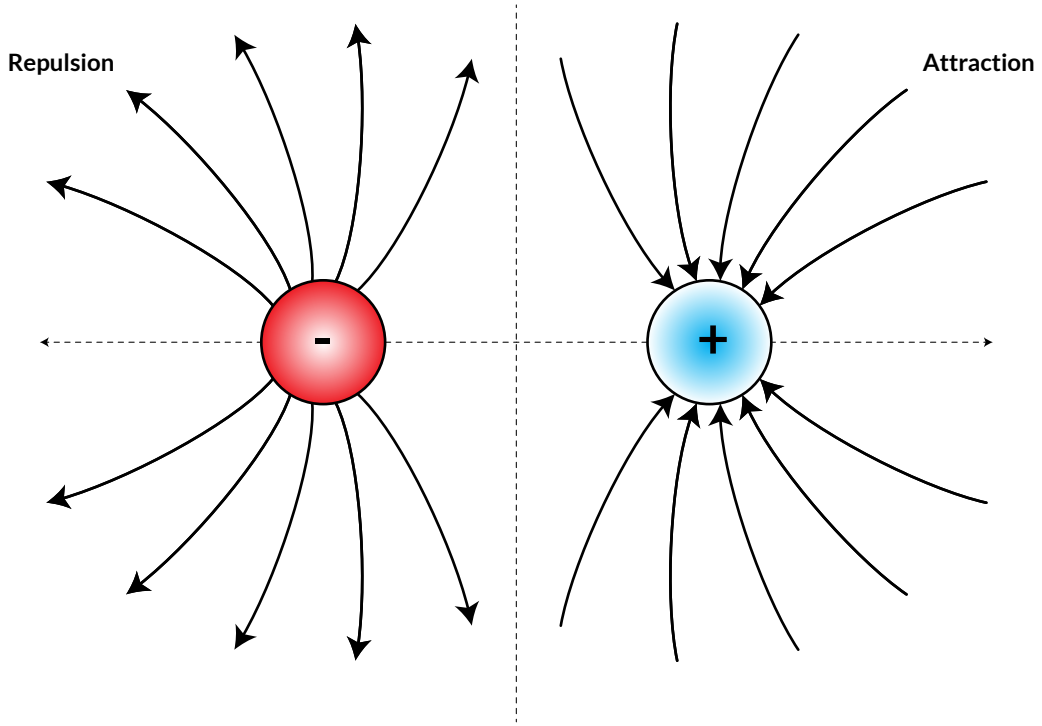
Daniel Shiffman diagram

Stigmergy

Stigmergy is a consensus social network mechanism of indirect coordination, through the environment, between agents or actions. The principle is that the trace left in the environment by an action stimulates the performance of a next action, by the same or a different agent. In that way, subsequent actions tend to reinforce and build on each other, leading to the spontaneous emergence of coherent, apparently systematic activity. - Wiki

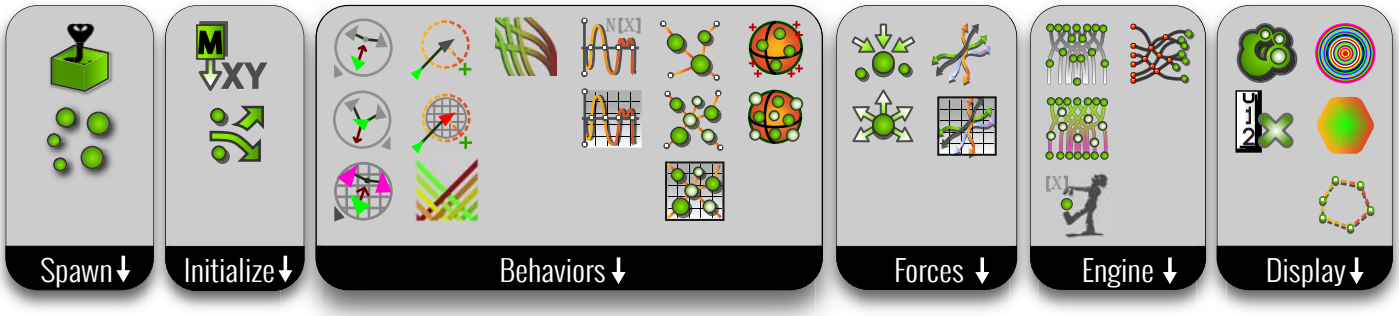
Forces

A force is a push or pull upon an object resulting from the object's interaction with another object. Whenever there is an interaction between two objects, there is a force upon each of the objects. When the interaction ceases, the two objects no longer experience the force. Forces only exist as a result of an interaction.



Culebra Components

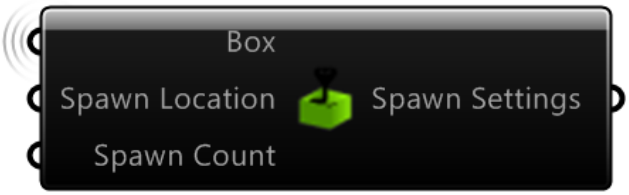
Culebra components are divided into six categories: Spawn Types, Initialize, Behaviors, Forces, Engine & Display

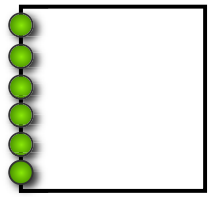


Spawn Types

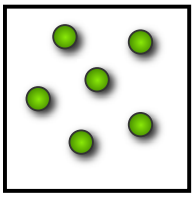
Box Spawn

Dynamic box spawn in 2D or 3D with two types of spawn options for each

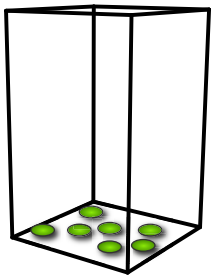




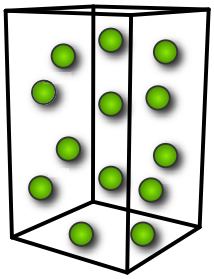
2D Aligned Spawn



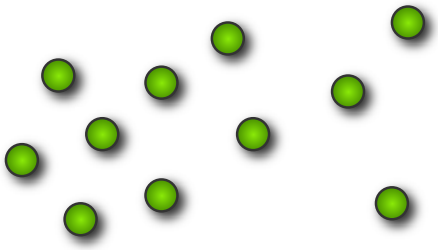
2D Random Fill



3D Random on Ground

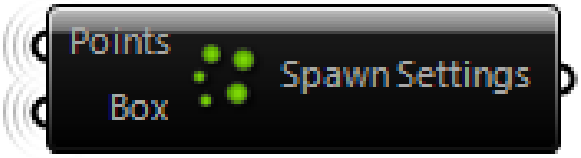


3D Random Fill



Point Spawn

User can specify specific points as starting locations for the creepers.



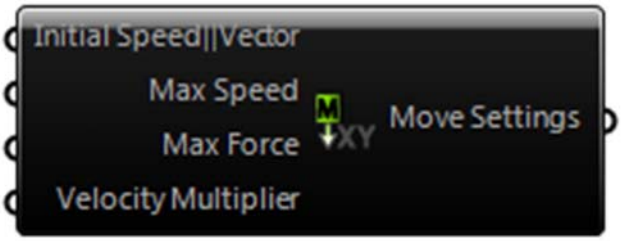
Init Settings
The initial settings specifying the spawn type, dimension and ability to constrain the creepers using a boundary.

Spawn Type For version 1.0 we have access to either box or point spawn.

Dimension User can specify 2D or 3D boundary and behaviors

Bounds If bounds is enabled the Flocking creepers will bounce off the boundary edges. In Perlin behavior they will die and respawn inside the boundary

Move Settings
Sends the move settings to the Creeper Engine



Initial Speed || Vector Input the initial speed vector or float value

Max Speed Input the max speed value allowed

Max Force Input the max force value allowed

Velocity Multiplier Input the velocity mutiplier

Connect Input a boolean toggle - True = Connect the heads (visualizes their search radius, DRAMATICALLY REDUCES PERFORMANCE) | False = Do not draw connectivity

View Angle Input a float value specifying the view angle each agent can see

Search Radius Input a float value specifying the distance each creeper can see

Align Value Input a float value specifying alignment (Steer towards the average heading of local flockmates) vector scale value

Separation Value Input a float value specifying separation (Steer to avoid crowding local flockmates) vector scale value

Cohesion Value Input a float value specifying cohesion (Steer to move toward the average position of local flockmates) vector scale value

Colored Mesh Input a color mesh to drive the flocking parameters

Map Alignment Input value specifying if you want the alignment value to be color driven

Map Separation Input value specifying if you want the separation value to be color driven

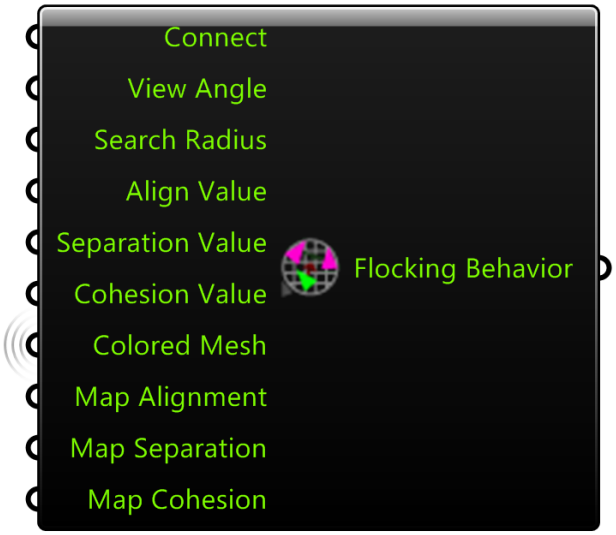
Map Cohesion Input value specifying if you want the cohesion value to be color driven

Flocking

Flocking behavior component

Flocking

Flocking Algorithm with image color sampling override for any flocking attributes and remaping of color values



Randomize Input value specifying if change value will be randomly selected from-change value to change value each frame

Add Heading Input value specifying if we want to add the heading

Change Input value specifying the incremented change value used to get the polar coordinates

Radius Input value specifying the radius for the wandering circle

Distance Input the distance for the wander circle, this is a projection value in the direction of the objects speed vector

Rotation Trigger This value is compared against each movement step. If rotationTrigger value > iteration count then we will reverse the change value

Type Input value specifying the type of Wandering (0 = Type A | 1 = Type B | 2 = Type C

Colored Mesh Input a color mesh to drive the flocking parameters

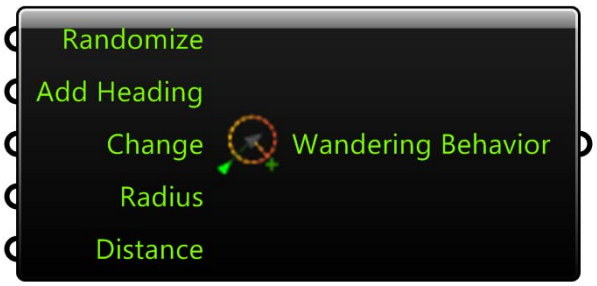
Map Change Input value specifying if you want the change value to be color driven

Map Radius Input value specifying if you want the radius value to be color driven

Map Distance Input value specifying if you want the distance value to be color driven

Wandering

2D Wandering Algorithm, Wandering is a type of random steering which has some long term order. Force Values from Move Settings have a strong effect on behavior



Weaving Wandering

Expanded 2D Wandering Algorithm using step triggers to create a weaving type movement 2D Wandering Algorithm, Wandering is a type of random steering which has some long term order. Force Values from Move Settings have a strong effect on behavior

Wandering

2D Wandering Algorithm with image color sampling override for any wandering attributes and remaping of color values, Wandering is a type of random steering which has some long term order. Force Values from Move Settings have a strong effect on behavior

View Angle Input a float value specifying the view angle each agent can see

Cohesion Mag Input float value for cohesion value to steers towards average positions

Cohesion Range Input float value for cohesion threshold, value within range will enable tailCohMag

Separation Mag Input float value for separation value to avoids crowding on trail

Separation Range Input float value for separation threshold, value within range will enable tailSep-Mag

Scale Input value specifying the noise scale

Strength Input value specifying the noise strength

Multiplier Input value to add a jitter type of movement

Velocity Input value specifying the noise velocity multiplier

Colored Mesh Input a color mesh to drive the flocking parameters

Map Scale Input value specifying if you want the scale value to be color driven

Map Strength Input value specifying if you want the strength value to be color driven

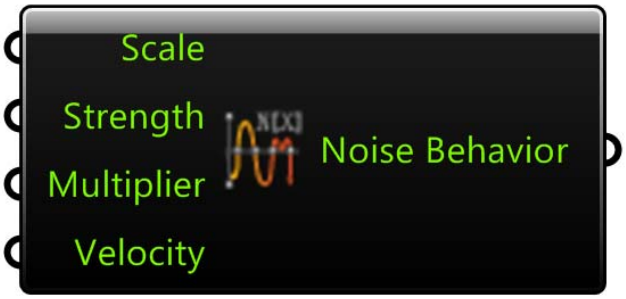
Map Multiplier Input value specifying if you want the multiplier value to be color driven

Stigmergy

2D/3D Trail Chasing Algorithm - Agents will chase agents trails. When using this algorithm in your main sketch use the overloaded move method, recommended values are move(6,100), in GH Trail Data TrailSet(6) Max Trails(100)

Noise

2D/3D Improved Perlin Noise



Noise

2D Improved Perlin Noise with image color sampling override for any behavior attribute

Polylines Input a list of polylines you want to follow, POLYLINE RESOLUTION IS IMPORTANT, KEEP AS LOW AS POSSIBLE

Threshold Input the distance threshold enabling agents to see shapes

Projection Distance Input the projection distance of point ahead on the path to seek

Radius Input the radius of the shapes

Trigger SPawn Input value specifying if creeper is now allowed to spawn any children objects stored

Max Children Input value specifying the maximum number of children each creeper can have, careful how large you make this number

Colored Mesh Input a color mesh to drive the path parameters

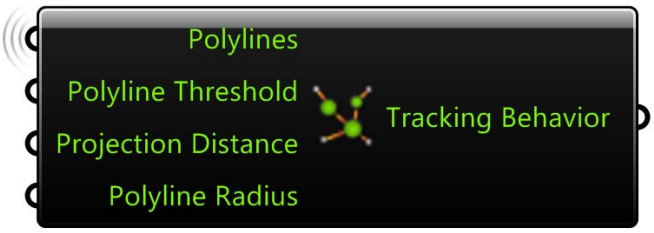
Map Threshold Input value specifying if you want the path threshold value to be color driven

Map Projection Input value specifying if you want the projection value to be color driven

Map Radius Input value specifying if you want the path radius value to be color driven

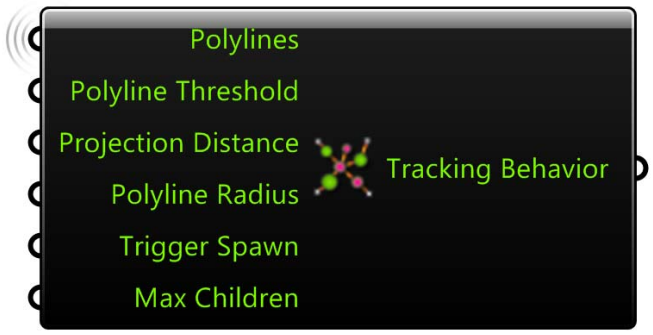
Multi Path Tracking

Multi Path Following Algorithm



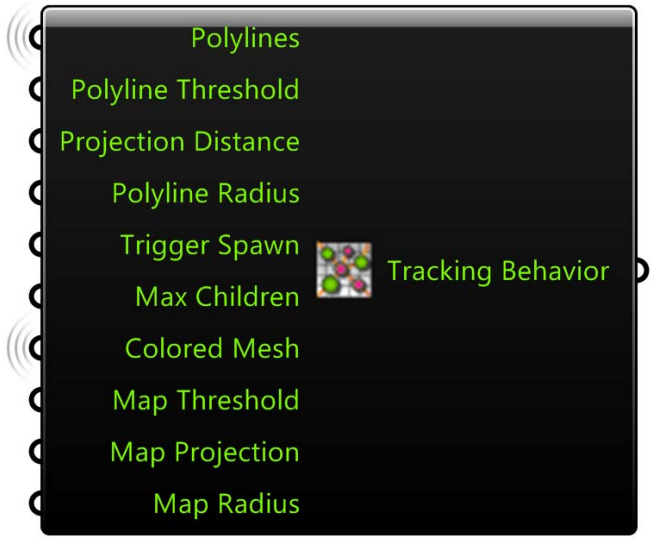
Multi Path Tracking II

MultiShape Path Following Algorithm capable of spawning children - see example files



Multi Path Tracking II

MultiShape Path Following Algorithm capable of spawning children with image color sampling override for any path attributes and remaping of color values - see example files



Mesh The mesh object to crawl on

Threshold Input the distance threshold, the min distance current position needs to be from mesh in order to move towards it

Projection Distance Input the amount to project the current location along the current speed

Multiplier Input the multiplier value

Trigger Spawn Input value specifying if creeper is now allowed to spawn any children objects stored

Max Children Input value specifying the maximum number of children each creeper can have, careful how large you make this number

Mesh Crawl

Mesh Crawling allows agent to move along a mesh object

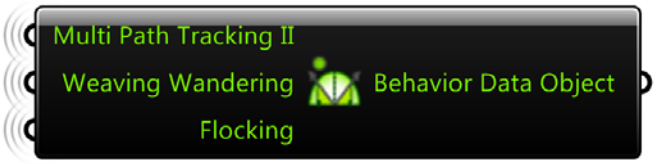
Mesh Crawl II

Mesh Crawling allows agent to move along a mesh object and is capable of spawning children

Dynamic Input Input any number of behaviors in any order to run nested behaviors. Input order is execution order. You can also re-arrange mid simulation

Controller

Behavior Merging Controller, you can add/remove/rearrange behaviors. The input order will be the behavior execution stack



Attraction
Attracts a object towards a set of targets

Targets Input a list of target points to attract to

Thresholds Input a list of threshold values, one for each target

Attraction Value Input a value specifying attraction, this is the magnitude

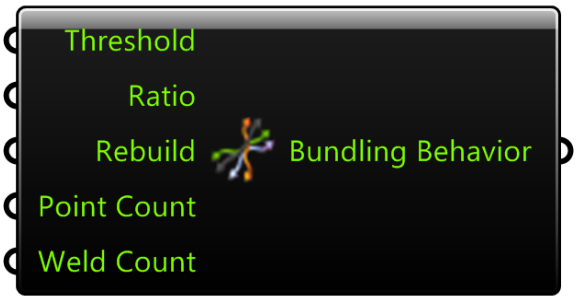
Max Attraction Input the maximum attraction value

Repulsion
Repels a object away from a set of targets

Repel Value Input a value specifying replusion, this is the magnitude

Max Repel Input the maximum repel value

Bundling
Settings for Self Organization of Curve Networks



Threhold Input float value specifying the distance threshold for self org

Ratio Input a float value specifying the distance each point can move per cycle

Rebuild Input a boolean toggle - True = Rebuild | False = Do not rebuild input curve

Point Count Input integer specifying rebuild curve point value

Weld Count Input integer value specifying how many points on each side of the curve you would like to weld

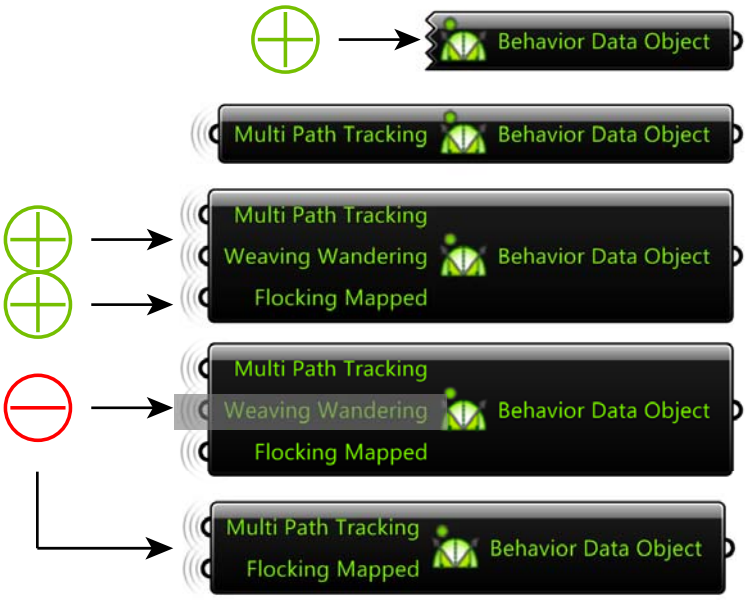
Bundling II
Settings for Self Organization of Curve Networks with image color sampling override for bundling attributes and remaping of color values

Colored Mesh Input a color mesh to drive the bundling parameters

Enable Color Input value specifying if you want the bundling threshold value to be color driven

Controller

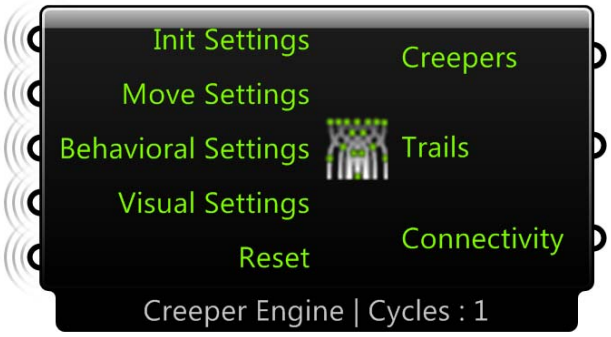
Behavior Merging Controller, you can add/remove/rearrange behaviors. The input order will be the behavior execution stack



Engines

Creeper Engine

Culebra Objects Engine



Init Settings Input the init settings output of Init component

Move Settings Input the move settings output from the Move component

Behavioral Settings Input the behavior settings output from the Controller component

Visual Settings Input the visual settings output of Viz component

Reset Input a button to reset the sim and reset all fields

Init Settings Input the init settings output of Init component

Move Settings Input the move settings output from the Move component

Behavioral Settings Input the behavior settings output from the Controller component

Visual Settings Input the visual settings output of Viz component

Child Move Settings Input the child move settings output from the Move component

Child Behavioral Settings Input the child behavior settings output from the Controller component

Child Visual Settings Input the child visual settings output of Viz component

Reset Input a button to reset the sim and reset all fields

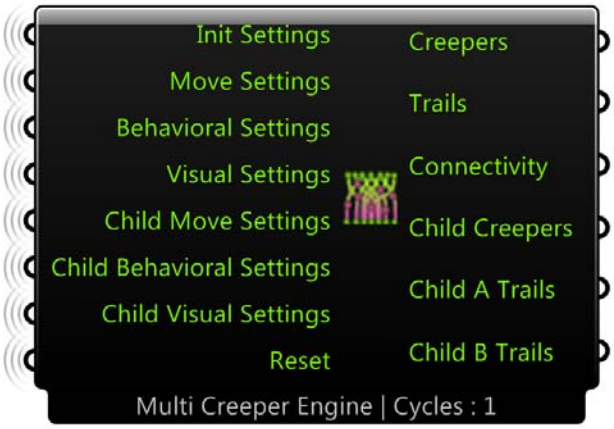
Iterations (Zombie) Input the number of steps you would like to run

Run (Zombie) Input a button to reset the sim and reset all fields

Curves (Bundling) Input polylines to bundle

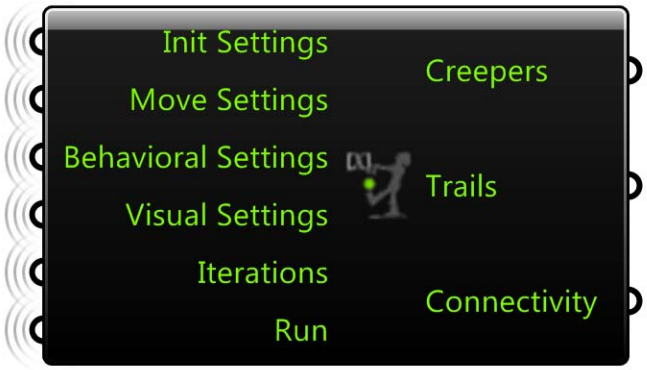
Creeper Engine

Culebra Multi Objects Engine



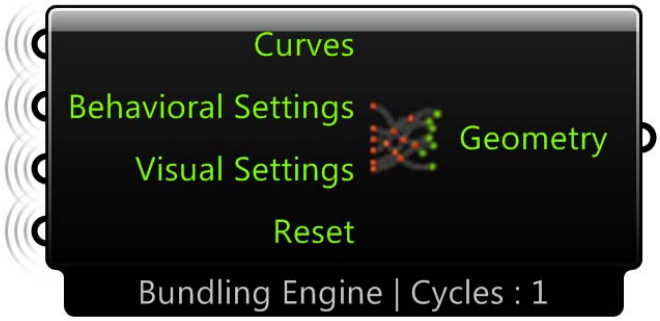
Creeper Zombie Engine

Culebra Object Zombie Engine



Bundling Engine

Engine for Self Organization of Curve Networks



Visual Data

Controls the visual settings for the Creeper Engine Outputs

Trail Data Input the Trail Data output from the Trail Data Component

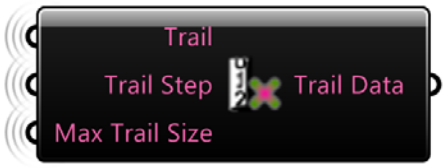
Color Data Input the Trail Color Data output from the Gradient Color Component

Apply Texture Input boolean specifying the application of texture as particle - WARNING VERY UNSTABLE

Display Mode Input an integer specifying the Display Mode (0 = Graphic | 1 = Geometry)

Trail Data

Controls the Trail Data for the Visual Settings Component



Trail Input a boolean toggle specifying trail visibility (True = On | False = Off)

Trail Step Input an integer value specifying the minimum amount of steps that must be taken before we store a trail vector (higher values increase performance)

Max Trail Size Input an integer value specifying maximum amount of allowable trail vectors per object (lower values increase performance)

Particle Texture Path
Input path to particle texture, a png file

Red Channel Input a domain component specifying the minimum and maximum floating point values for the red channel

Green Channel Input a domain component specifying the minimum and maximum floating point values for the green channel

Blue Channel Input a domain component specifying the minimum and maximum floating point values for the blue channel

Min Thickness Input the polyline minimum thickness

Max Thickness Input the polyline maximum thickness

Color Input a color to use for the graphical polyline drawing

Dotted Input a boolean value to specify polyline dotting (True = Solid | False = Dotted)

Thickness Input the polyline thickness

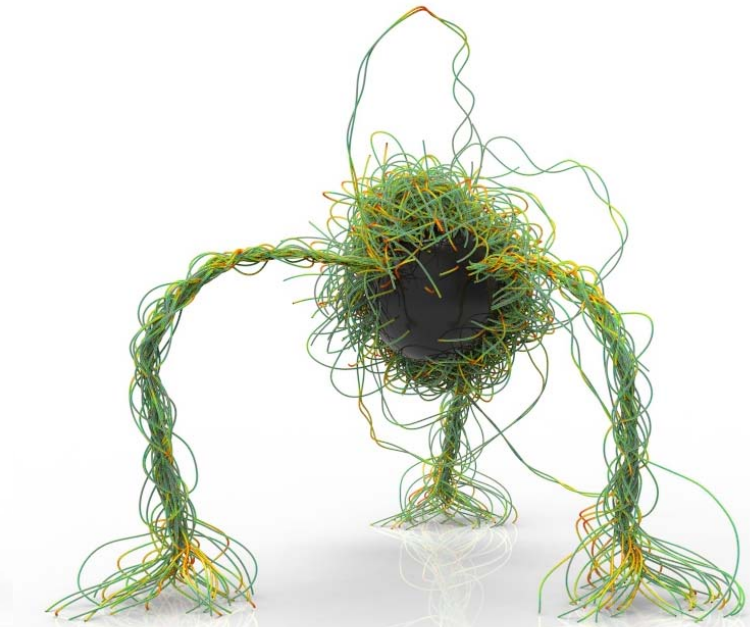
Gradient Color
Controls the Gradient Color trail options for the Visual Settings Component

Graphic Polyline
Controls the Graphic Polyline Color trail options for the Visual Settings Component

Disco Colors
Controls the Disco Color trail options for the Visual Settings Component

Swarn Mesh Behavior

Attraction no Walk
Causes gliding around the mesh



Attraction no Walk
Causes gliding around the mesh

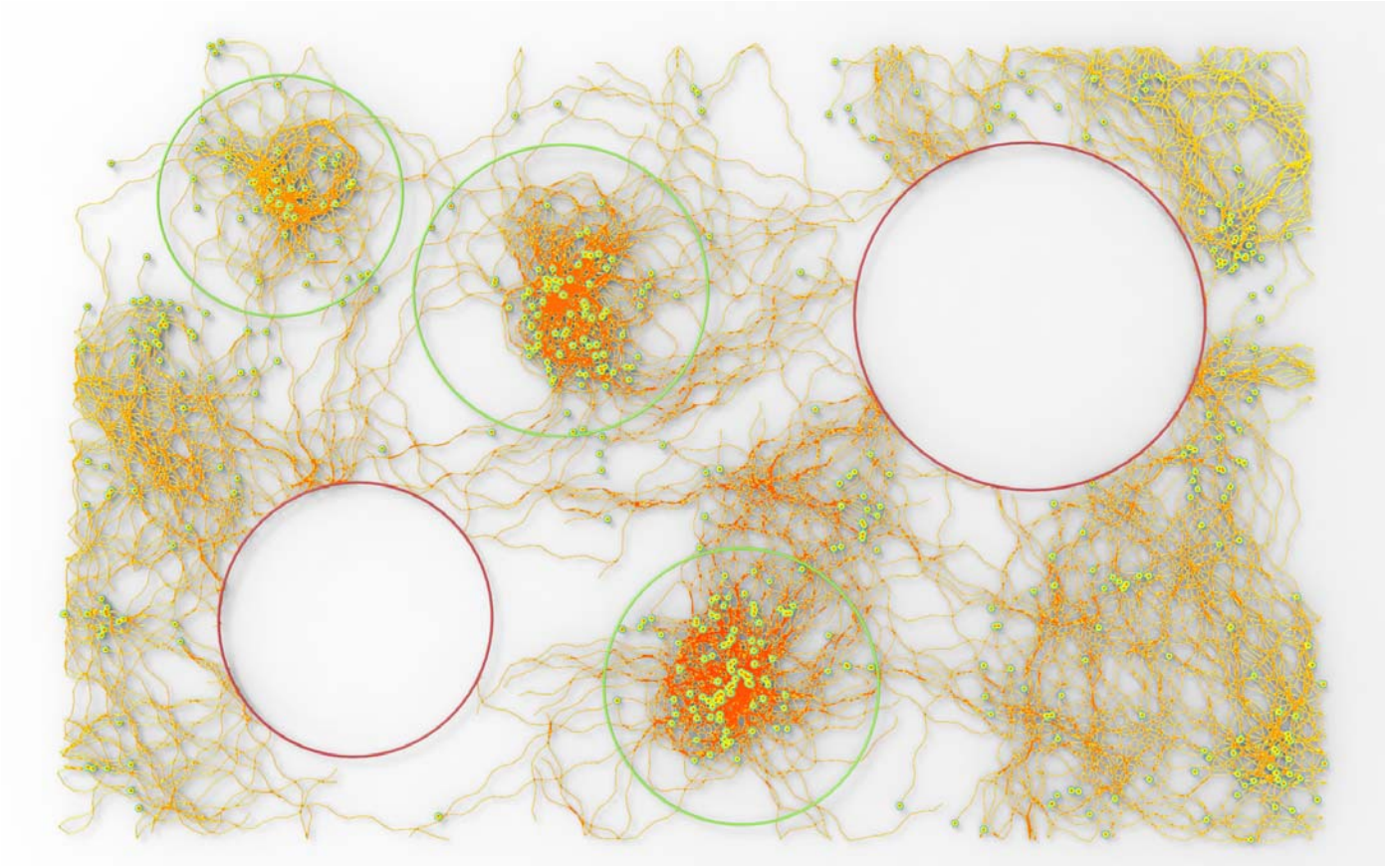
MultiAttraction Walk
Walks along the mesh, whichever mesh its closest to

Repulsion
Moves away from object if within thresh

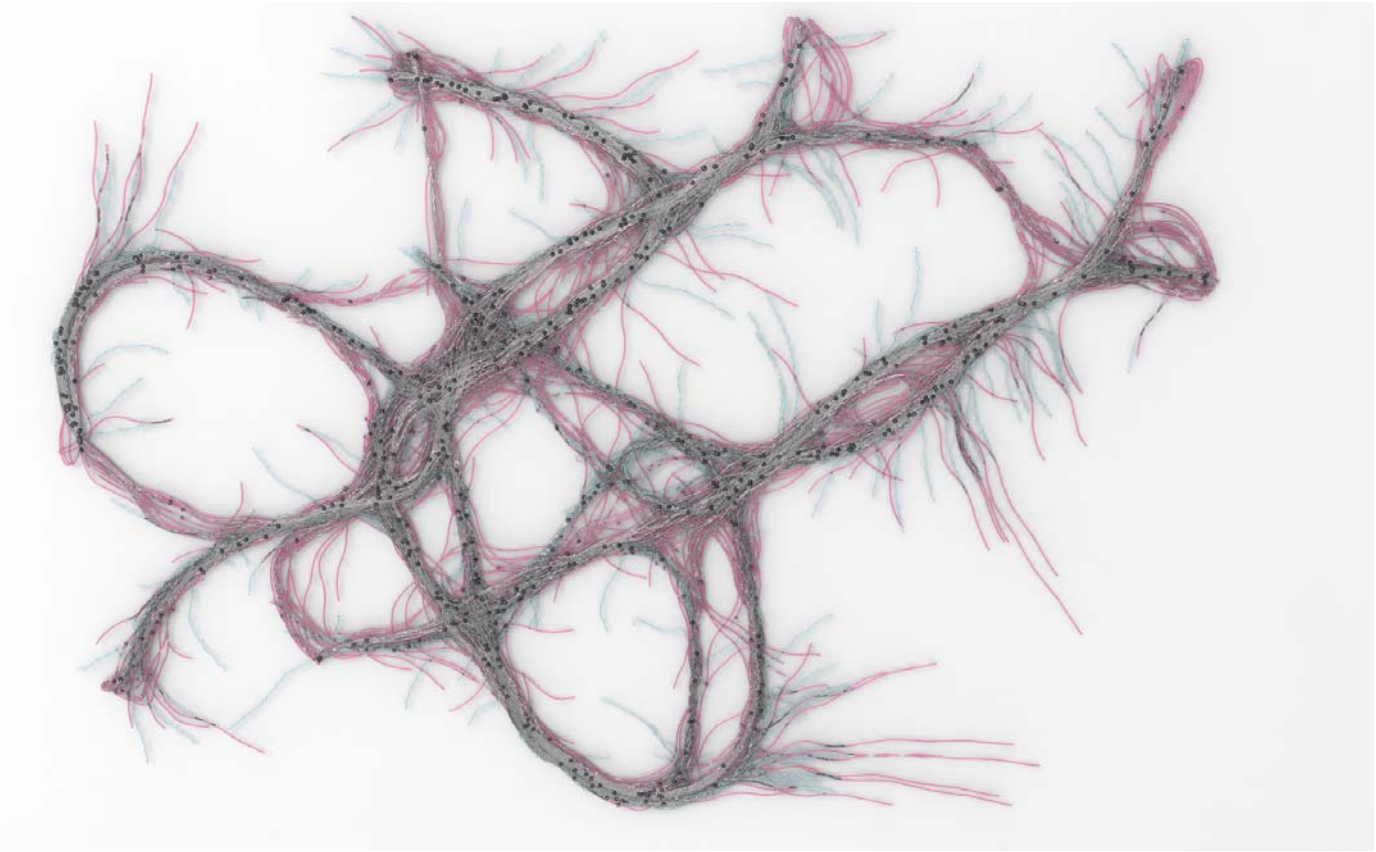
Mesh Crawl II

3D Perlin Noise

Attraction + Repulsion



Multi Object Path Following




complicitMatter

CULEBRA

v 2.0 Creepy things For Grasshopper

Contact

PERSONAL

luis@complicitMatter.com

www.complicitMatter.com
www.computationalMatter.com

MODE LAB CONTACT

lquinones@modelab.is

http://modelab.is/