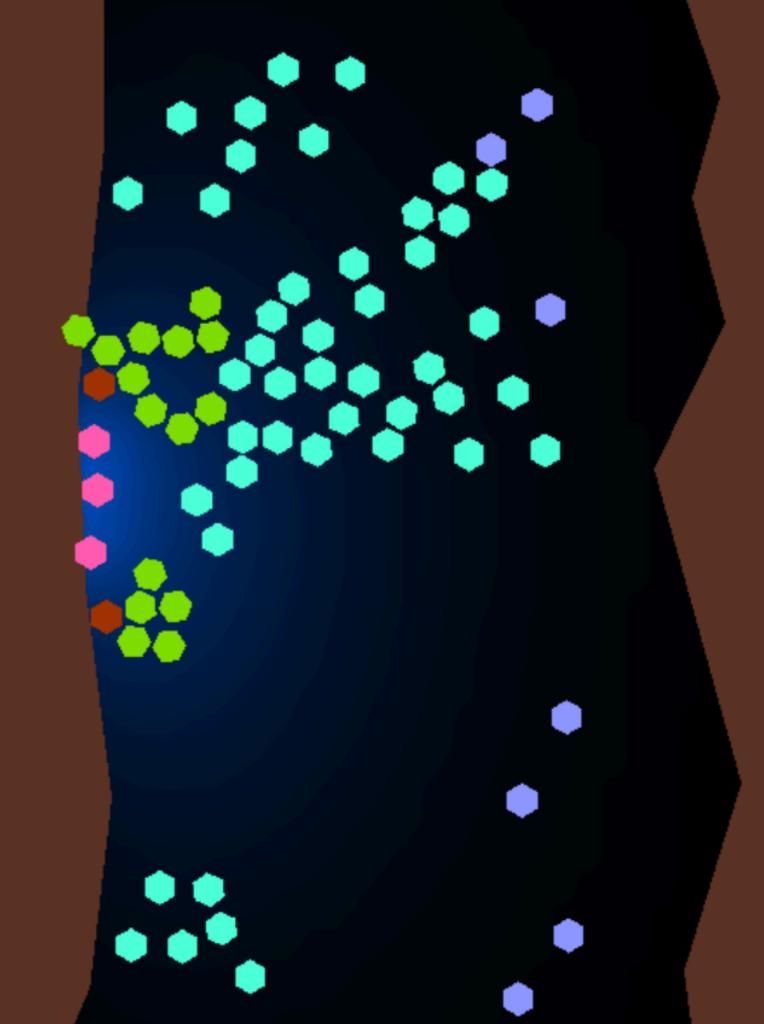
Pascal BALLET

A Godot addon to create 2D or 3D swarms, flocks and multi-agent systems



A brief

INTRODUCTION



The main node of Behavior Tree For Groups (BTFG) represents a Godot group (a disk inside a square) completed with a trunk, to draw a tree.

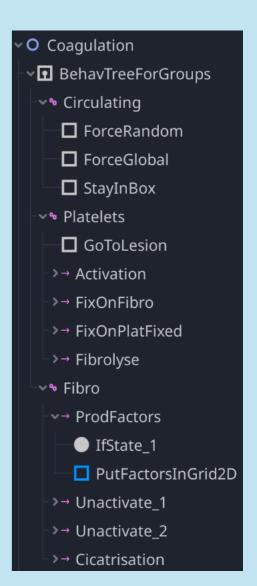
BEHAVIOR TREE FOR GROUPS

BTFG is an addon for Godot Engine that enables you to create swarms, flocks and multiagent systems using a single behavior tree.

You can easily design collective behaviors without any code by adding groups to your nodes.

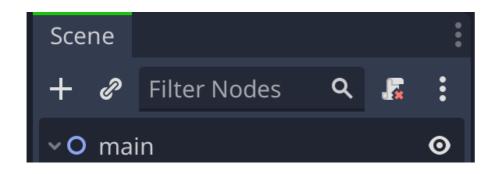
Several examples, in 2D or 3D, are available when installing the add-on in Godot: test them to understand how they work and what can be done.

When imported into your project, add groups to your nodes then create a new BTFG that works onto the groups of your choice.



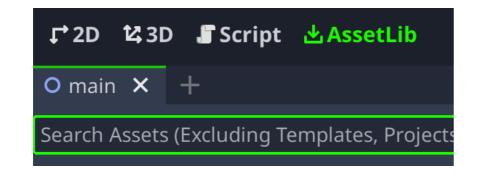
Simulation of blood coagulation made with BTFG.

SIMPLE 2D TUTORIAL



First, create a new 2D scene

- save it as main.tscn



Import...

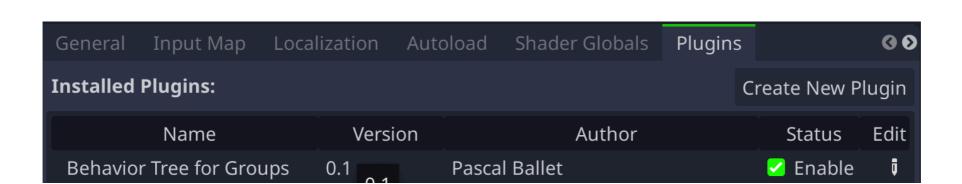
Plugins...

Addon download

- From the AssetLib tab of Godot, search, select, then download the Behavior Tree For Groups addon.

Plugin activation

Do not forget to *Enable* it from the *Plugins* section.



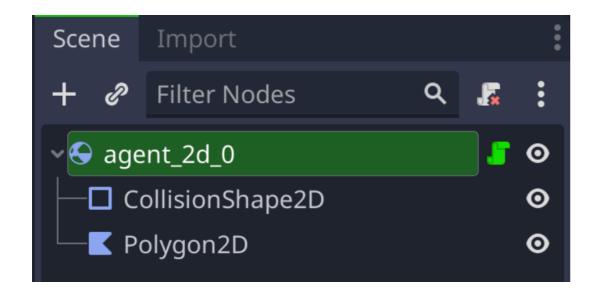
SIMPLE 2D TUTORIAL



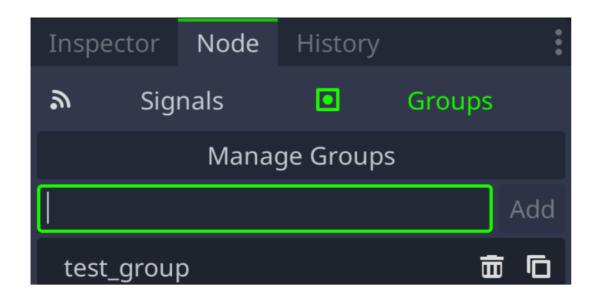
Now, create a new Agent2D

- Click the + button in your scene tree and
- Select NewAgent2D
- A file called *agent_2d_0.tscn* is automatically generated into your file system

NB: this node is NOT added to your scene tree, but is just a convenient way to create an agent, which is a *RigidBody2D* with its *MeshInstance* and *CollisionMesh*. If you prefer, you can also create your own *RigidBody2D*.



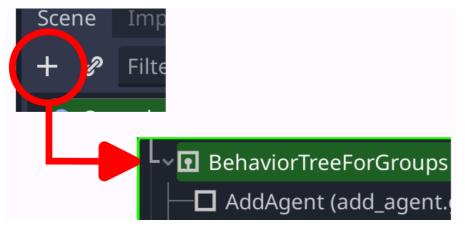
- Open the agent_2d_0.tscn automatically generated, and
- select the *RigidBody* at the root of this scene.

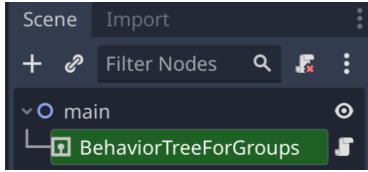


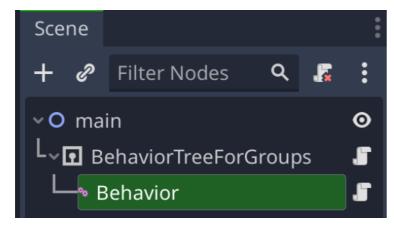
- In the inspector, go to the Node Tab
- then go to Group
- and add a new group called test_group

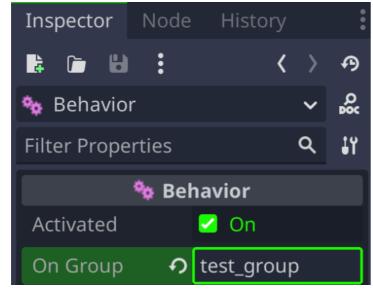
Save the agent_2d_0.tscn scene

And go back to your main scene









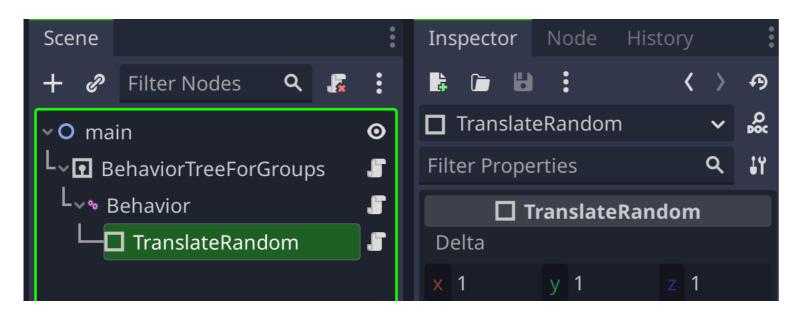
Then, create a simple behavior tree

- Click the *plus button* in your *main* scene, then select the node *BehaviorTreeForGroups*

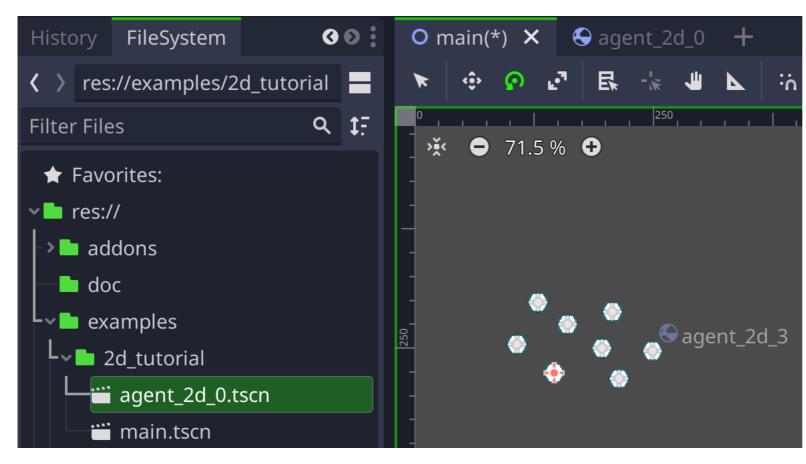
- Then, add a new child node called *Behavior*

- In the inspector of the newly created node Behavior, in the Group property, write test_group. That means all the nodes having the *Group test_group* will be treated.

Add an Action in the Behavior



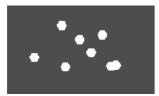
- As child of the node Behavior, add a new node called TranslateRandom



Finally, add many agent_2d

- by drag & drop from the file agent_2d.tscn to your main scene
- place 8 of them in your main scene







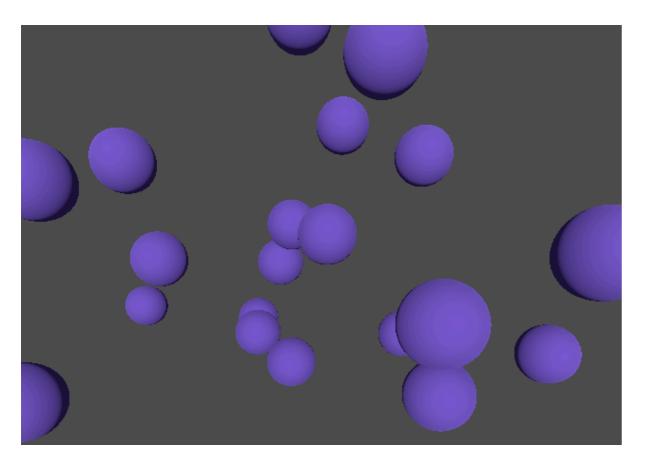
- ▼ BehaviorTreeForGroups (behavior_tree_for_groups.gd)
- ─☐ AddAgent (add_agent.gd)
- AddSpringOnContact (add_spring_on_contact.gd)
- 😽 Behavior (behavior.gd)
- ☐ ChangeColor (change_color.gd)
- CloneAgent (clone_agent.gd)
- DataPlus (data_plus.gd)
- ☐ DataSet (data_set.gd)
- Delete (delete.gd)
- -? Fallback (fallback.gd)
- ForceAngularZ (force_angular_z.gd)
- ForceForward (force_forward.gd)
- ForceGlobal (force_global.gd)
- ForceRadial (force_radial.gd)
- ☐ ForceRandom (force_random.gd)
- ForceToGridValue (force_to_grid_value.gd)
- IfCollision (if_collision.gd)
- IfDataEquals (if_data_equals.gd)
- IfDataInf (if_data_inf.gd)
- IfDataSup (if_data_sup.gd)
- IfGrid2DValueInf (if_grid_2d_value_inf.gd)
- IfGrid2DValueSup (if_grid_2d_value_sup.gd)
- IfInBox (if_in_box.gd)
- IfMouseClick (if_mouse_click.gd)
- IfNoContact (if_no_contact.gd)
- IfProba (if_proba.gd)
- IfStateEquals (if_state_equals.gd)
- IfStepBetween (if_step_between.gd)
- ─<mark>□</mark> NewAgent2D (new_agent_2d.gd)
- NewAgent3D (new_agent_3d.gd)
- NewGrid2D (new_grid_2d.gd)

Execute your program

- all the agents having the group *test_group* will move at random
- for example, you can change the speed of the translation by changing the default value Delta to 5 in the TranslateRandom node

Conclusion

Many other nodes exists to create many different behaviors



SIMPLE 3D TUTORIAL

Very simple

- Replace 2d by 3d in the previous example!
- And do not forget to put a *camera* and a *light* before you execute your program