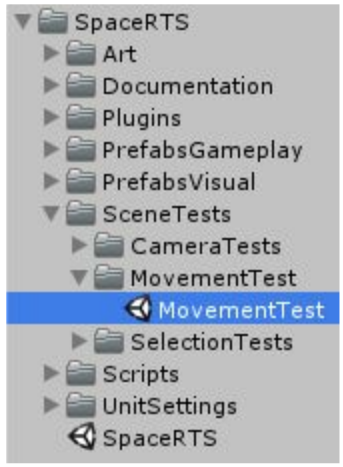
RTS Starter Kit

Units Movement

Introduction

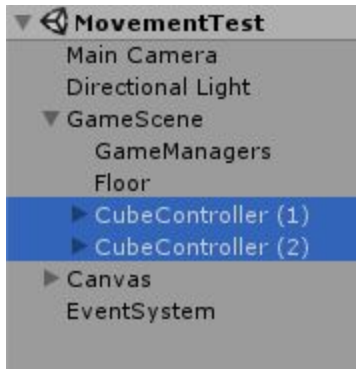


In this document, we are going to explain how it's builded, how to configure and how are the communication between the UI to handle the input, the Group Controller (that handles the destination position for each unit and maintains the units formation) and the units that we need to be moved.

In order to do the explanation more visual and simple we are going to analyze the scene called "MovementTest" (included in the package).

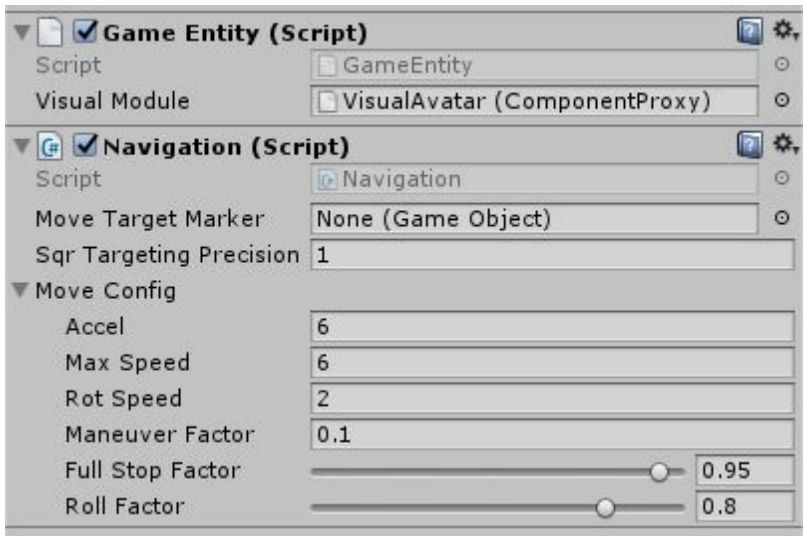
The scene goal is to control 2 spheres and move it through the platform with a simple right click on the desired destination. This will produce a smooth movement of the spheres maintaining a strict formation.

Units Configuration



Here we cover how to correct setup the units to achieve a proper and fluid movement. in our “MovementTest” scene we can find out four units able to be controlled (highlighted in the image from the right). if we select one then we can see how it’s configured.

Of course, first of all we have the GameEntity component which defines our VisualModule and also required for the correct configuration of our game unit. but next to it we have the Navigation component.



(How it looks the CubeController (1) GameObject)

Navigation
Handles the unit’s movement. Here we have some parameters to talk about:

Move Target Marker: Handles the feedback for the movement order. This is not a requirement so in our test scene isn’t used at all but you can see it in action in the “SpaceRTS - Demo” scene.
Sqr Targeting Precision: How far from the targeting position it’s considered as target reached? the value is exposed as a square distance.

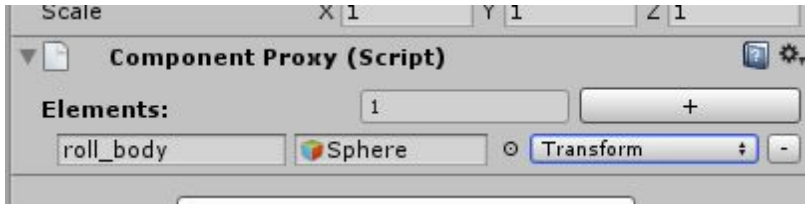
Under the Move config we have:
Accel: of course, the unit’s acceleration.
Max Speed: The top speed for this unit
Rot Speed: The top speed for the rotation
Maneuver Factor: This is tricky, Saying that the MaxSpeed is the top speed when the unit is moving in straight line; when the unit is turning, how much slowdown will be produced

according to the current turning angle? That's the maneuver factor. A zero value will be no reduce at all the speed when turning.
Full Stop Factor: This factor will be applied when the unit is not traying to accelerate producing a steady slow down of speed until a full stop is reached. a value of zero will produce an automatic full stop (not very appealing) and a value of 1 will produce no reduction of speed, so the unit will have a sted movement in the current direction (like the effect produced by no friction in the space)
Roll Factor: We have the possibility to produce a roll effect in the unit to achieve a hovercraft effect when the unit is turning. This parameter changes how much roll will be produced in the transform specified by the VisualProxy as "roll_body" according with the current rotation. (See "Visual Module" in this document).

Move order Feedback
The "MovementTest" scene doesn't use any feedback at all for the movement but we can see the "Space RTS - Demo" to see how it works. In the Hierarchy panel find out the game object under "GameMap/GameEntities/HumanShip". Here in the Navigation component we find a Move Target Marker setted. if you click it then the referenced MovementMarker GameObject will be remarked. As you can see it's just a quad with a circle texture that will be showed at the target position to remark the movement destination.

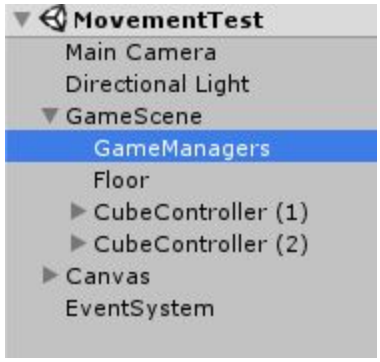
Visual Module
In our VisualModule (called VisualAvatar in the "Movement Test" scene) there is no property configured in the ComponentProxy. That is true. No parameter is needed. but we can use an optional property to achieve a hovercraft effect if we sets a property called "roll_body" and referencing a Transform in our avatar that will be rolled to achieve that effect.

It should look like this:



Of course, in our current scene will have no visual effect because we are controlling a sphere. but you can see the complete effect under the "Space RTS - Demo" scene. see how the ships changes its direction and a hovercraft effect is produced.

Group Controller System



Only this system component is required to handle the move of large groups of GameEntities. In our MovementTest scene it's located under the GameScene GameObject in the GameObject called "GameManagers".

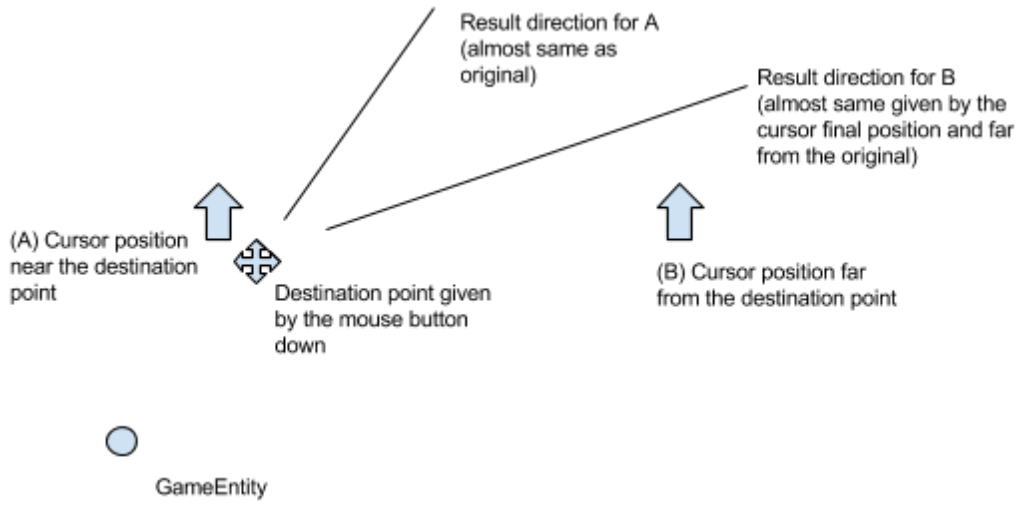
How it work

Basically exposes 3 methods that handles the movement orders for the group.

MoveHandlerBegin() that is called when the user presses an input button over the screen. is here where the destination point is established.

MoveHandlerDrag() optional method called during the users dragging to allow the user to change the final direction for the GameEntities. Also refresh the partial destination for each navigation that will be moved. By default the direction will be facing from the original position to the destination but as long as the cursor leaves the original destination position the direction will be more close with the current cursor position. How far needs the cursor be from the destination position to achive the perfect direction to the mouse cursor? that can be manipulated by the directionPrecision component parameter exposed in the editor.

MoveHandlerEnd() Assigns the established position and direction to all the navigation belonging to the group.



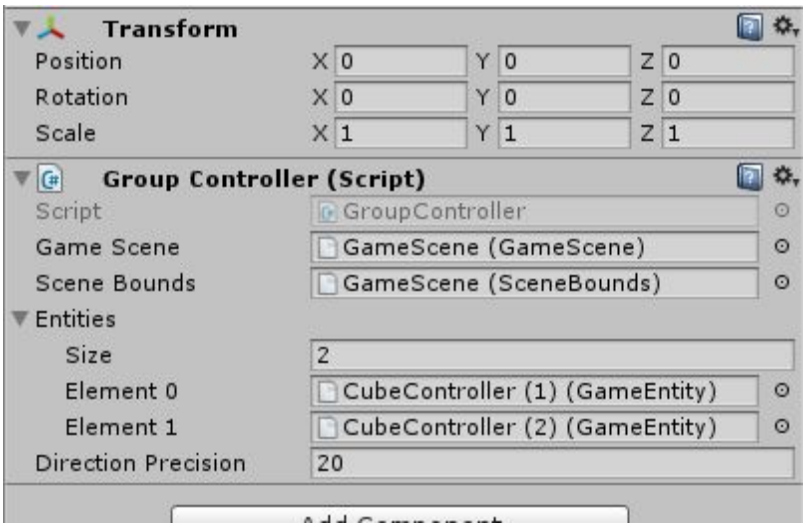
(Two examples showing how the final direction for the GameEntity can change according to how far is the mouse button up from the position when the mouse button down was executed)

Configuration

Requires the GameScene but also the SceneBounds where this component takes the cursor world position when the MoveHandler methods are called.

The list of GameEntities that will be controlled by this component. Make sure that each of these GameEntities has the Navigation Component attached.

Finally, the Direction Precision that tells how far needs to be the cursor button up from the cursor button down in order to actually change the direction as explained in the last topic.



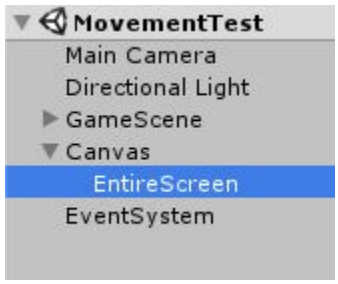
(Here is how it looks the GameManagers GameObject)

UI layer

Keeps track of the user input (assuring that only works when in correct focus) and derive that actions to the input controller according to the current game state.

Some basic setup

Below the Canvas we have the panel that controls the focus over the game screen. In this case, will handle the pointer button down and pointer button up.



This is achieved thanks to the Event Trigger attached that calls to the corresponding GroupController methods.