Space RTS Starter Kit

Introduction

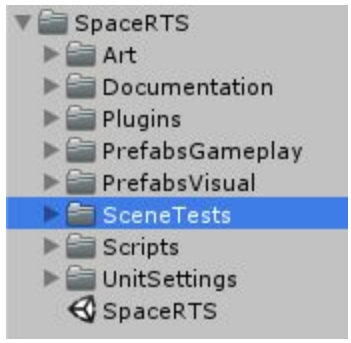
Welcome

Hi, Thanks for your interest in my product! This is a small guide for SpaceRTS - Starter Kit. Hope you find it useful to build a quick and beautiful prototype for your project or as a starting point for a full scale Space RTS game.

for the moment i have the next features implemented:

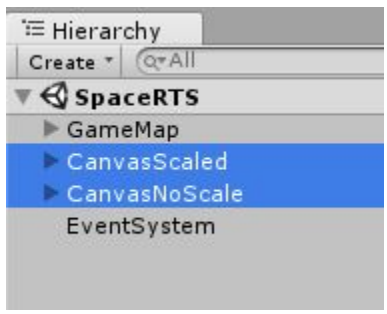
- Mouse hovering feedback
- Point and click selection
- Box drag selection
- Multiple selection
- Smart Selection by unit type (TODO)
- Customizable RTS Camera
- Camera pan with middle mouse button
- Camera pan through screen borders
- Unit customization
- Smooth ship movement
- Ship building through Structures
- Structure building through ships

but i'm thinking to implement some more in the future, maybe starting with pathfinding integration? let me know your feedback!



I recommend to checkout the "SceneTests" folder where you can find different individual test for my mayor systems (you can explore it for better understanding of each of it) and of course the "SpaceRTS" scene where you'll find the full featured project.

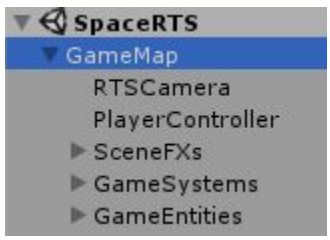
Scene Structure



About the UI; There are two canvas. why two?

CanvasScaled will hold all the ui that need to be scaled when the screen changes. that means that almost all the game's UI will be holded there (buttons, screens, panels, settings menu, etc).

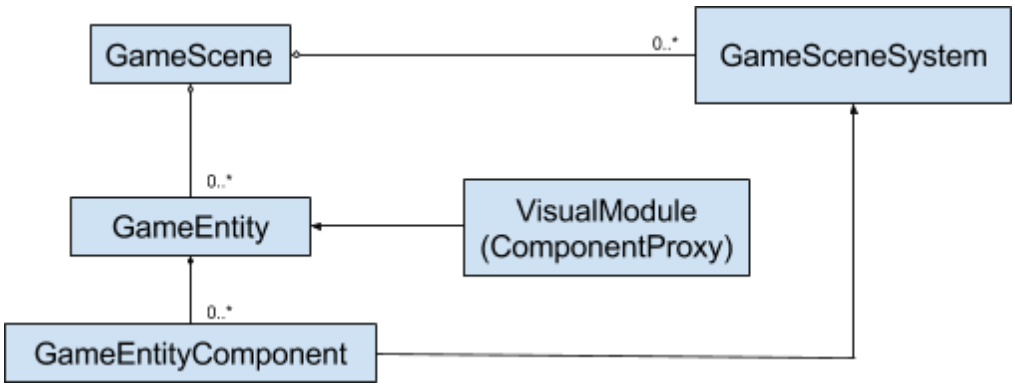
CanvasNoScale will hold all the ui that never should scale, that is the for example the selection box that needs absolute position from the cursor to set its screen location, width and height.



GameMap will hold the mayor systems and entities of the rts game and 3 mayor containers holding them. SceneFXs, GameSystems and GameEntities.

How it works?

The core game system diagram should look something like this:



There is a main GameScene at the root that contains all the GameSceneSystems that registers at it. Then there is a bunch of GameEntities (our ships, structures, and all the things that interact through the game and needs to be found between each others). Finally we have the GameEntityComponent that will access to their needed system through the entity and then through the GameScene.

Another thing is the visual module, stored in the GameEntity. If the GameEntity is part of the controller that gives the interaction with the world, the VisualModule will be the bridge to the visual avatar of our entity. there it's the character, the ship, the starbase mesh, animation, etc.

Why i choose this aproach?

Because i want to separate the gameplay from the visuals. in that way the artist can work in the model and can have it's own gameplay implementation for testing. A lot of times we have the Gameplay GameObject that has a lot of components interacting with a lot of systems.

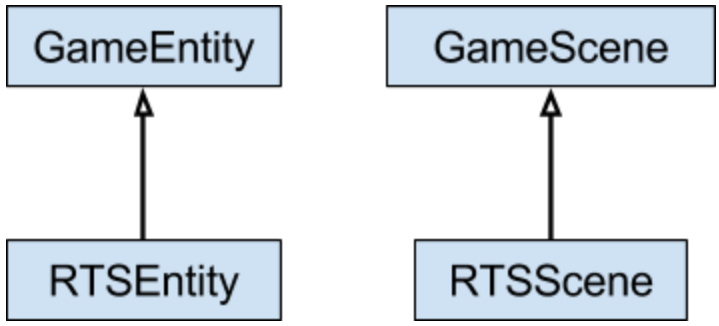
When the artist needs to test the model or some basic interactions the only way its testing it with the actual game. With this aproach we can construct a different scene with a GameEntity with just a bunch of gameplay component, just the scentials for the test porpouse or maybe creating a few new ones just to test.

That way we can have our game scene with a ship that can be destroyed by other ships shooting at it and at the same time we can have a test sceen with the same model ship but we can trigger the destruction when the user clicks on it. better don't you think?

In my test scenes i've separated each major feature from the others and that minimum features are still working without the others.

SpaceRTS Structure

Some inheritance diagrams for the SpaceRTS:



RTSScene

Wil read the input and act according to the game state. Is the MainController for the game, but must be act as a central point and delegate specific responsibilities for the GameSceneSystems.

GameSceneSystems

These are some of the game scene systems we can found in our SpaceRTS demo.

- **SelectionSystem and SelectionInput:** Handles the hover, highlight and selection of the units in our game.
- **GroupController:** controls the movement handler to obtain a destination position and moves the selected ships in a specific formation to the that point.
- **GhostBuilder:** Helper that allows us to select a build location for our structures.

RTSEntity

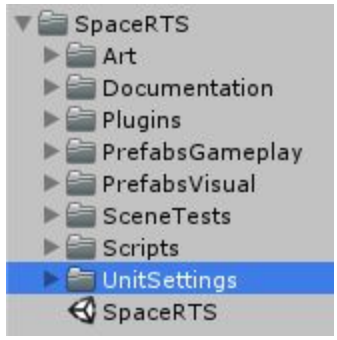
Holds the RTSPlayer (PlayerController) in charge of keep tracking of the user's entities and the UnitConfig that is the setup data for our unit, it's like the genetic code of the unit. The UnitConfig will be better explained in the next topic.

GameEntityComponents

These are some of the gameplay components used in the game:

- **Selectable:** Gives to the GameEntity the possibility of been hovered, highlighted and selected.
- **Navigation:** With this component the unit can move through the map.
- **Builder:** With this, the GameEntity can build other units.

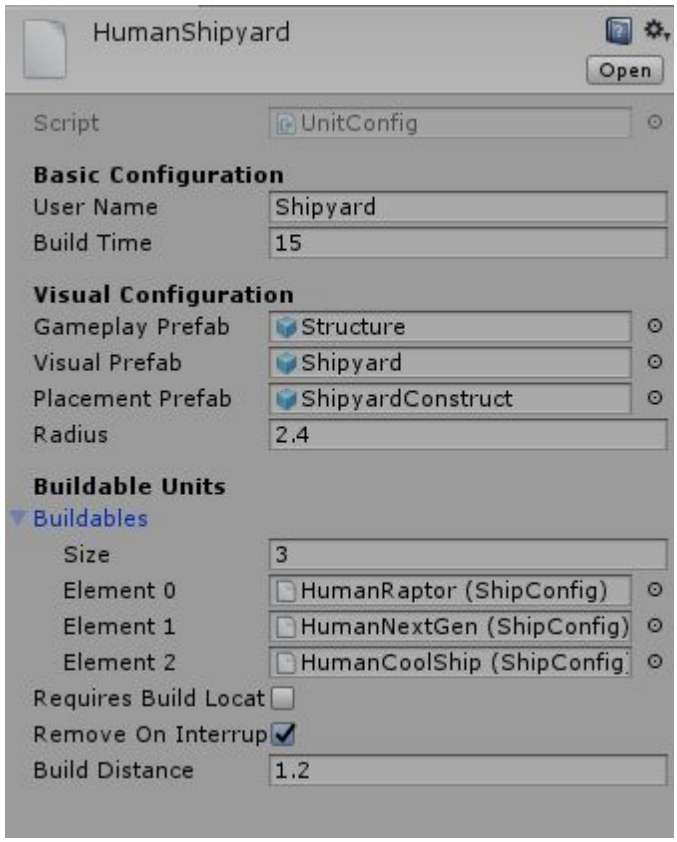
UnitConfig



Located under the UnitSettings folder we can find all the current unit configurations. There are 2 kinds on configs, UnitConfig and ShipConfig.

UnitConfig provides the base configuration to all units, contains the reference to the required gameplay and visual prefabs (that will be needed when we want to spawn a new unit), the placement prefab will be used by the ghost controller to select a placement for the structure to be built.

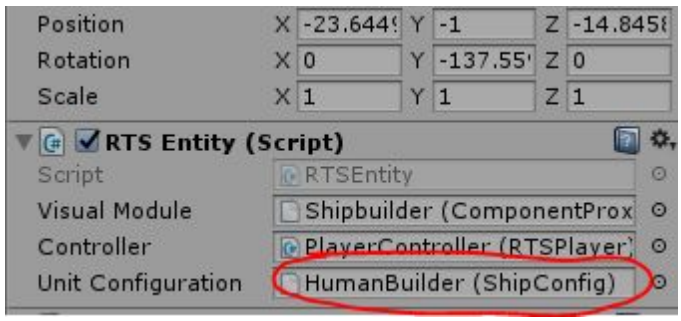
Radius represents the boundings of the unit (a volume thing)



Buildables is a list of UnitConfigs that this entity it's capable to build. in the example, the shipyard is capable to build 3 kinds of ships (HumanRaptor, HumanNextGen, HumanCoolShip)

RequiresBuildLocation tells that a placement will be required to build all this units (in this case no location is required because all the ships are built inside of the shipyard)

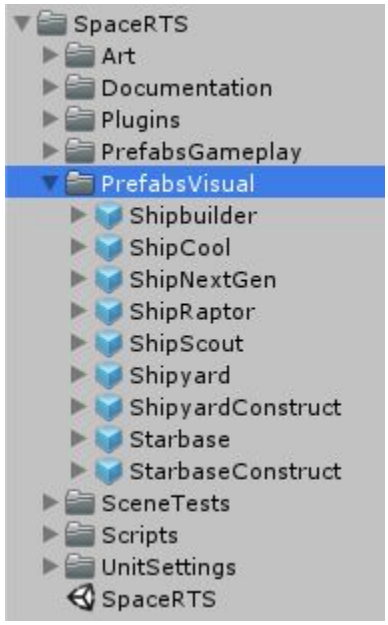
RemoveOnInterrupt is applied for all the units that are built inside of others, because if the build is interrupted means to be cancelled and never will be completed, that is not the case for the structures, they are built in a location and when interrupted the structure stays in place and can be completed later.



Build Distance means how far from the build placement needs to be this builder to start the production.

MovementData Present in the ShipConfig type, gives the navigation setup needed for each ship, here we can setup different velocities for each kind of ship among other things.

Visual Modules



Inside the PrefabsVisual folder we can find all the VisualModules already configured, there are examples of Ships and structures. The ShipyardConstruct prefab is the semitransparent green version of the shipyard that will be used by the ghost controller as feedback when the user its trying to set a build location for the unit.

The only rule that matters here is: at the root object we need to place a ComponentProxy.

Remember: This object doesn't contain any gameplay components inside. just visual components, like automatic rotation components, MeshRenderers, Colliders, Animators and so on.

ComponentProxy

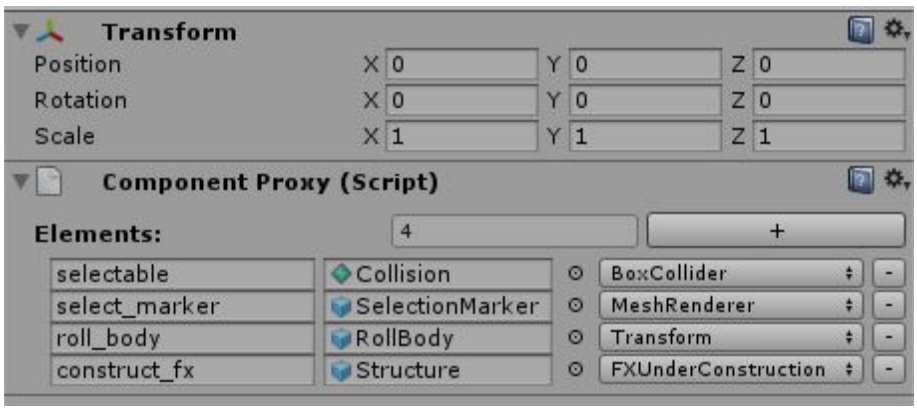
Is the nexus between the gameplay object and the visual object made by the artist. Contains references to all the important components in the prefab and a dictionary for an easy access by code to that components.

How to setup?

Easy, add elements with the plus button. drag the child object that contains the component that you want to store, once the object is setted, select from the list from the right the name of the component, at the left, give it identifiable name.

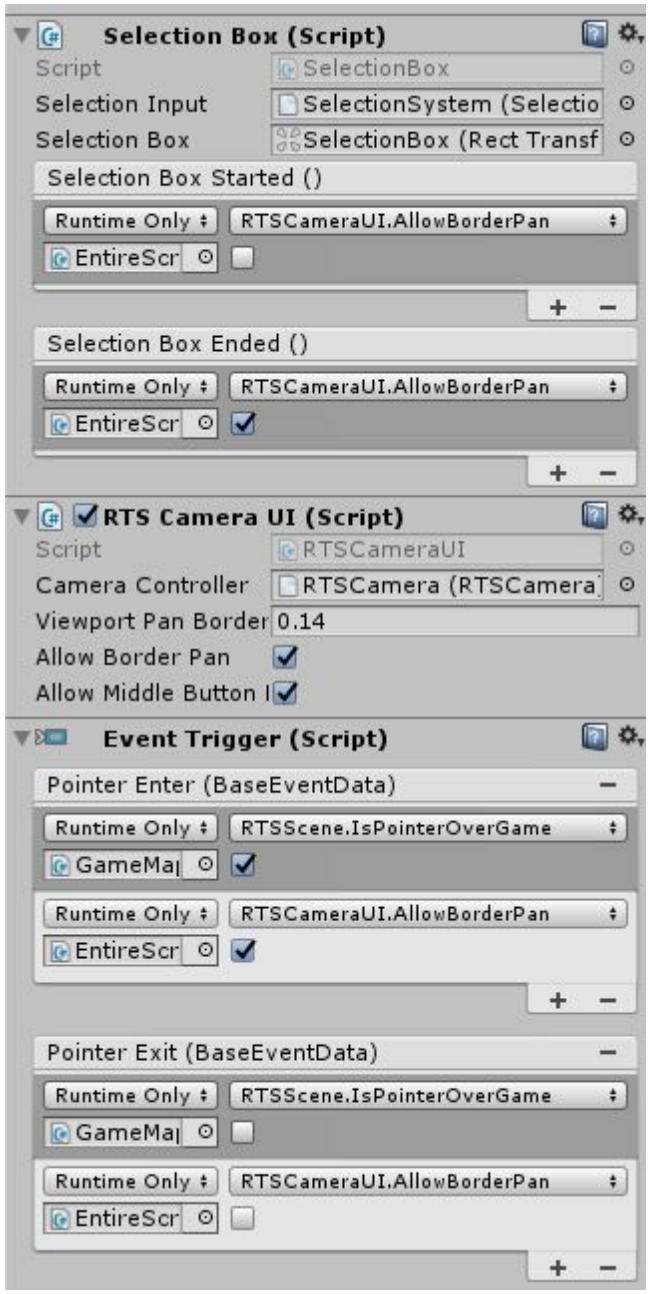
Important: The name aren't random, the names are been used by the components in the gameplay side of the entity. so, for example "selectable" will be used by the Selectable component, "select_marker" will be used by the SelectionFeedback component, and so on.

That components requires the specific names, so please check in the documentation of each component the requested name. i'm working in a best way to expose that names in each component editor.



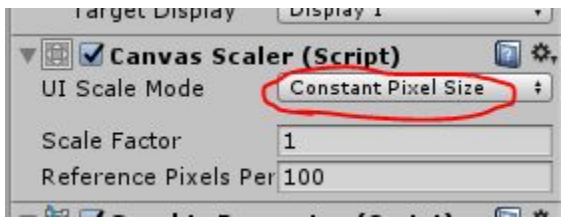
UILayer

As i said before, there are two canvas:



CanvasNoScale

Handles the SelectionBox (Input that later will be transmited to the SelectionInput), the RTSCameraUI (Checks the cursor over the screen borders, and middle button camera panning). Also checks if the gameplay client area isn't covered by any other panel (like the builders ui) because it's supposed that for example no units must be selected when it's covered. So, also informs to the RTSScene if it's covered or not (Done by the Event Trigger component)



CanvasScaled

Here we have all the panels that must scale according to the screen size. In this case i'm refered to the HelpPanel (A simple toggle button displaying a plain text area showing the input references) and the UnitSelectUI (Panel that will be shown when a unit build capable is selected).

What's next?

Well, that would depend on all your feedback. I'm thinking that the next thing to do it's to integrate rigidbodies to improve the navigation and prevent the ships moving inside others or inside structures. Also a good pathfinding algorithm to dodge structures along its path. Let me know all your feedback and questions at:

nullpointer2017@gmail.com

for a online code reference:

http://nullpointer.getforge.io/

for a full online description of the different modules and how it works:

Units Movement Guide

RTS Camera Guide

Units Selection Guide

Thanks again, and hope to get your feedback!