How to convert legacy from Swarm Assault game and create own mission

Tutorial

1. Swarm Assault map creation

First we have to make a map in Swarm Assault game. It can be either auto-generated or made from scratch.

If you choose making a map from scratch, I highly recommend using this custom map editor made by **jmscreator**:

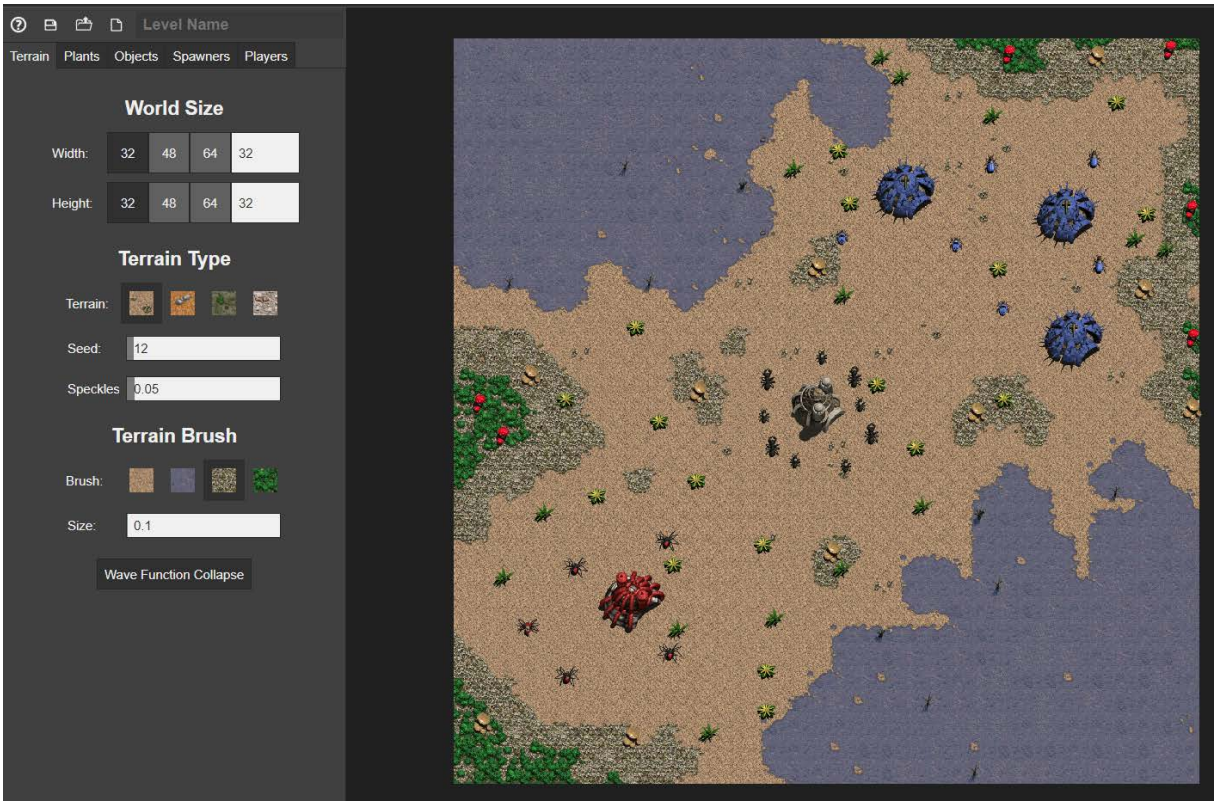
Link to the YouTube video:

https://www.youtube.com/watch?v=ceJtYrRmqKo

Link to the editor itself

https://sites.google.com/view/swarm-assault-deluxe-editor/home

For the purpose of this tutorial, I created a custom simple map using the above editor:

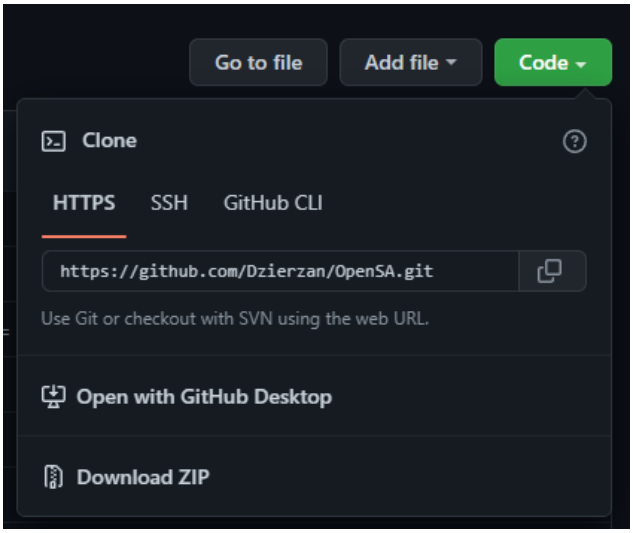


2. Download the master branch of OpenSA

In order to import maps from the vanilla game, it is required to get developer version of OpenSA. You can it from here:

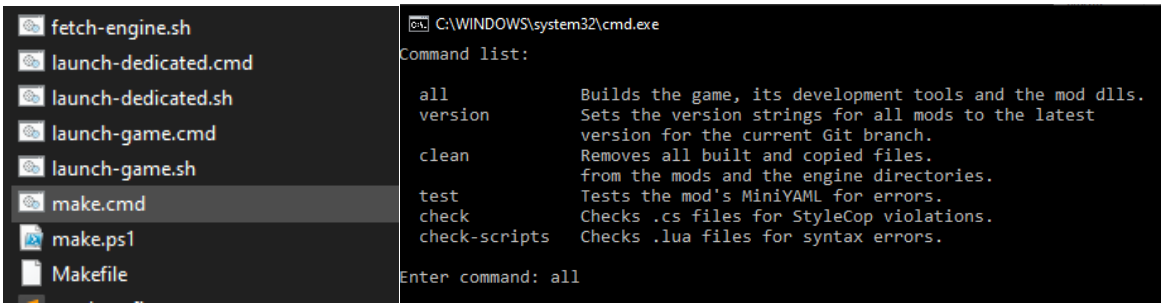
https://github.com/Dzierzan/OpenSA

You can either fork the whole project or download the latest ZIP. In this tutorial we will use the ZIP one.



3. Compile the developer version

In order to use developer version, it needs to be complied. To do so, double click file named "**make.cmd**". Once the new windows shows up, type "**all**" without the quotation marks and press enter.



It is very likely that the complication process will fail. Mostly due to missing a lot of required tools or NET missing. Read carefully what tools are required and get them from the Internet. Everything required is free of charge.

For additional information please visit this website:

https://github.com/OpenRA/OpenRA/blob/bleed/INSTALL.md

4. Import a map

If the complication process was successful, we can finally import a Swarm Assault map.

Put your custom map in ~/OpenSA/engine/ folder.

Double click file “utility.cmd” located in ~/OpenSA/ folder (NOT in **engine** folder).

The formula for map import is as follows:

--import-sa-map MAP.LVL TILESET

Where:

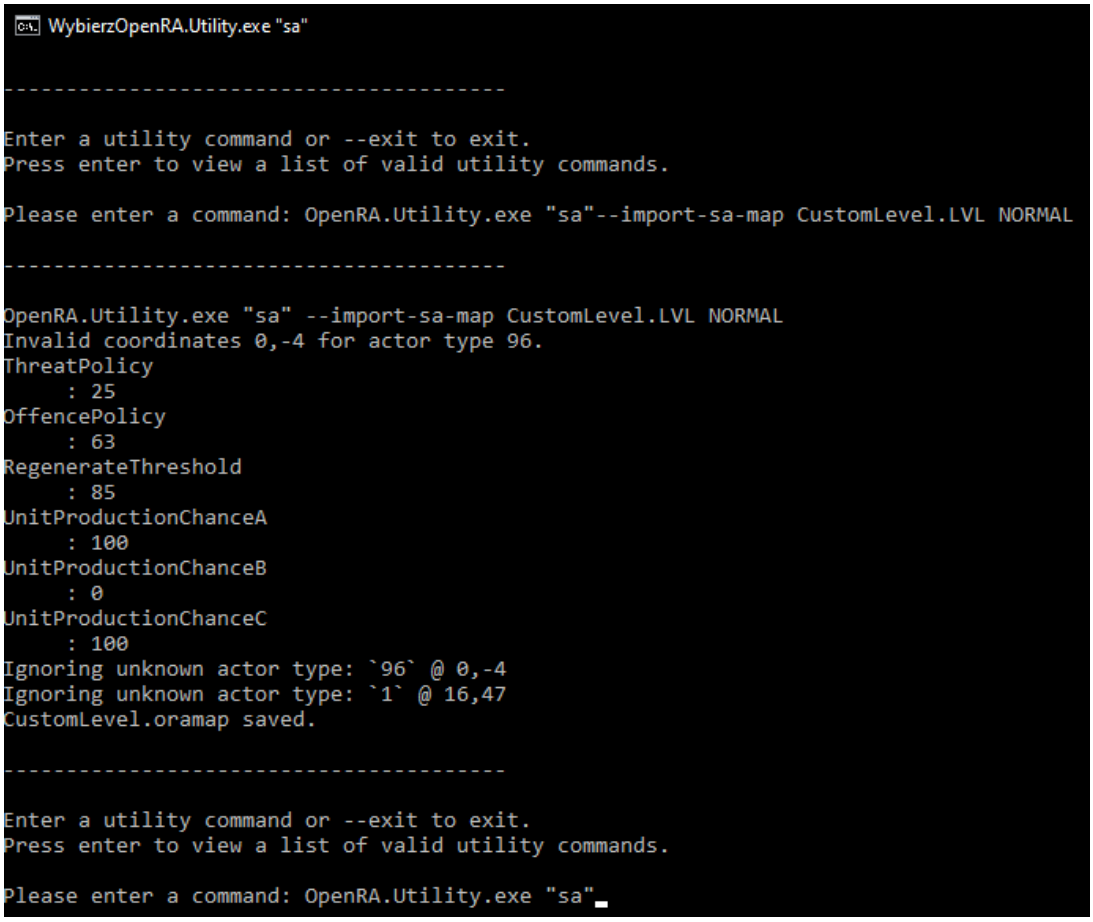
MAP.LVL – the name of your custom map

TILESET – tileset used for the map. Four options are available:

NORMAL, SWAMP, DESERT, CANDY

Since my map used the default tileset, NORMAL parameter will be used.

Below you can see an example of properly imported map. If it’s done correctly, OpenSA compatible map will be generated in ~/OpenSA/engine/ folder. In my case it’s named “CustomLevel.oramap”.



5. Extract your map in maps folder

The imported map is not ready yet for playing. The importer doesn't import information who is the player, who is the enemy, what creep units to spawn and so on. We will have to provide such information.

Oramap extension basically means the map is packed and it can be extracted using any extracting tool (WinZip, WinRAR or 7-Zip). It recommended to extract your custom map in support directory where the game keeps the saved settings, maps, replays, logs and mod assets. The default location is as follows, but one can choose to move it to an arbitrary location by passing an Engine.SupportDir argument to the Game.exe

Windows:

\Users\<Username>\AppData\Roaming\OpenRA\maps\sa\{DEV_VERSION}

macOS:

/Users/<username>/Library/ApplicationSupport/OpenRA/maps/sa/{DEV_VERSION}

GNU/Linux:

~/.config/openra/maps/sa/{DEV_VERSION}

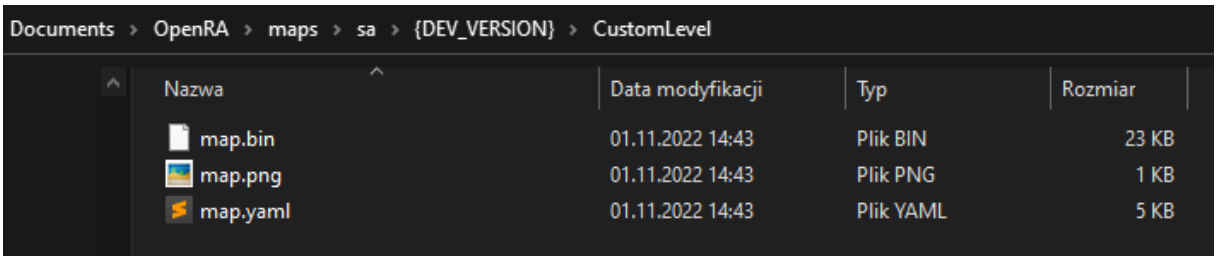
Older releases (before OpenRA playtest-20190825) used different locations, which newer versions may continue to use in some circumstances:

Windows:

\Users\<Username>\My Documents\OpenRA\maps\sa\{DEV_VERSION}

GNU/Linux:

/home/<username>/.openra/maps/sa/{DEV_VERSION}



6. Setting up map.yaml

Open file **map.yaml** with notepad or any other text tool. Right now the most important section which we need is “Players”. It tells us about set factions. Right now it doesn’t tell us who the enemy and who is the player. In our map, Red (Spiders) will be the Player and Blue (Beetles) will be enemy AI. Below I showed how it should be set up.

I highly recommend reading other example map.yaml files for other missions.

Default	After changes
Players: PlayerReference@Neutral: Name: Neutral OwnsWorld: True NonCombatant: True Faction: Random PlayerReference@Creeps: Name: Creeps NonCombatant: True Faction: Random PlayerReference@Spiders: Name: Spiders Faction: spiders Color: CA3131 PlayerReference@Beetles: Name: Beetles Faction: beetles Color: 606EA5	Players: PlayerReference@Neutral: Name: Neutral OwnsWorld: True NonCombatant: True Faction: Random PlayerReference@Creeps: Name: Creeps NonCombatant: True Faction: Random Enemies: Spiders, Beetles PlayerReference@Spiders: Name: Spiders Faction: spiders Color: CA3131 Playable: True Required: True LockFaction: True LockColor: True LockSpawn: True LockTeam: True Enemies: Beetles, Creeps PlayerReference@Beetles: Name: Beetles Faction: beetles Color: 606EA5 Bot: beetles-ai Enemies: Spiders, Creeps

We can also change that, so our map will be shown in missions lobby.

Default	After changes
Visibility: Lobby Categories: Conquest	Visibility: MissionSelector Categories: Campaign

At the bottom there’s also “Rules” section. We need add additional rules like that:

Rules: sa|rules/campaign.yaml, sa|rules/ai-campaign.yaml, rules.yaml

7. Setting up rules.yaml

Now we need setup what AI can build and if we want creep units such as Dragon Fly.

We create a file named “rules.txt” in our custom level folder and then rename its extension to yaml or we can take “rules.yaml” from other mission and edit it properly. The choice is yours. Below I showed it should look like for our map, in the next page I will explain that better:

```
World:
	MissionData:
		Briefing: Hello world!
		StartVideo: sa|bits/videos/TeamLogoRed.vqa
		WinVideo: sa|bits/videos/TeamWinRed.vqa

World:
	PirateSpawner:
		SpawnInterval: 1000, 1500
		InitialSpawnDelay: 1000, 1500
	CreepFlyerSpawner@dragonfly:
		SpawnInterval: 1000, 1500
		InitialSpawnDelay: 1000, 1500
	PlantSpawner@NORMAL:
		Minimum: 1
		Maximum: 15
		SpawnInterval: 375, 750
		InitialSpawnDelay: 375, 750
Player:
	-GrantConditionOnBotOwner@BeetlesAI:
	ExternalCondition@BeeltesAI:
		Condition: enable-beetles-ai
	UnitBuilderBotModule@BeetlesAI:
		UnitsToBuild:
			ants_light: 50
			#ants_medium: 0
			ants_heavy: 100
			beetles_light: 50
			#beetles_medium: 0
			beetles_heavy: 100
			scorpions_light: 50
			#scorpions_medium: 0
			scorpions_heavy: 100
			spiders_light: 50
			#spiders_medium: 0
			spiders_heavy: 100
			wasps_light: 50
			#wasps_medium: 0
			wasps_heavy: 100
	SquadManagerBotModule@BeetlesAI:
		SquadSize: 10
		SquadSizeRandomBonus: 5
		AirUnitsTypes:
		ProtectionTypes:
```

Parameter	Description
Briefing	Here you can set information about your map which will be displayed in the missions lobby.
StartVideo	A video which will play as soon as you start a mission
WinVideo	A video which will play as soon as you win a mission.
PirateSpawner	Black ants spawner. If it has dash in front of the name, it's disabled, without it, it's enabled: PirateSpawner: --- enabled -PirateSpawner: --- disabled
CreepFlyerSpawner@dragonfly	Trait which spawn creep flyer units. In this case, it's Dragonfly.
PlantSpawner@NORMAL:	Trait which spawn creep plant units. In this case these are Popcorn and Venus plants.
SpawnInterval	Average time (ticks) between creep spawn.
InitialSpawnDelay	Delay (in ticks) before the first creep spawns.
Minimum (only for PlantSpawner)	Minimum allowed amount of spawned actors.
Maximum (only for PlantSpawner)	Maximum allowed amount of spawned actors.

Parameter	Description
-GrantConditionOnBotOwner@AI	It needs to be like that.
ExternalCondition@AI: Condition: enable-X-ai	Condition for AI to be enabled.
UnitBuilderBotModule@X_AI: UnitsToBuild:	What units AI should build and what is the probability of producing specific unit, so 50 means 50%. # in front of actor means the production of this type of unit is disabled. In theory 0% should also disable production, but for some unknown reason, it doesn't. So commenting out an actor is a makeshift solution.
SquadManagerBotModule@X_AI: SquadSize: n SquadSizeRandomBonus: n	The squad size of enemy group.

8. Setting up lua script.

Now our custom map needs very basic lua script for AI activation. You can grab it from any other mission or create your own file named “script.txt” and then rename it to “script.lua”. The choice is yours.

Here’s the code for our map:

```
BotDelay =
{
	easy = 15,
	normal = 6,
	hard = 3,
	veryhard = 1
}

WorldLoaded = function()
	Camera.Position = Actor47.CenterPosition
	Beetles = Player.GetPlayer("Beetles")
	Trigger.AfterDelay(DateTime.Seconds(BotDelay[Map.LobbyOption("difficulty")]),
function()
		Beetles.GrantCondition("enable-beetles-ai")
	end)
end
```

Parameter	Description
BotDelay	How much time in seconds must pass before AI actives.
WorldLoaded = function()	Main function
Camera.Position	The default position of the camera. It is set on actor 47. We can get this information either from **map.yaml** or OpenSA editor itself if we select any actor.
Beetles = Player.GetPlayer("Beetles")	Setting faction AI
Trigger.AfterDelay	Trigger an even after X amount of time
Beetles.GrantCondition("enable-beetles-ai")	Grant “enable-beetles-ai” condition so Beetles AI can be activated.

9. Finish

After setting basic rules, AI and lua script, the mission should be in playable state.

The new custom mission should appear at the bottom of list in missions.

As I mentioned before, I highly recommend reading rules of other maps and use that as a reference point.