We now plan to add the new voxel ore to game.
this will require 4 *.sbc files which we will create in SEMods/MyMod/Data

These files can be named anything the tags the xml files use to begin, are whats important.
but to keep them named in a manner we can recognize them, we should choose a naming convention for our mod now.
We will name this one Tutorial, so this means we will preface all our files with Tutorial_, lets begin.

Our first file is VoxelMaterial.sbc, we call ours Tutorial_VoxelMaterial.sbc

It will look like this

```
<?xml version="1.0"?>
<Definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <VoxelMaterials>
<VoxelMaterial xsi:type="MyObjectBuilder_Dx11VoxelMaterialDefinition">
    <Id>
      <TypeId>VoxelMaterialDefinition</TypeId>
      <SubtypeId>Thorite</SubtypeId>
    </Id>
    <SpawnsInAsteroids>true</SpawnsInAsteroids>
          <SpawnsInPlanets>true</SpawnsInPlanets>
    <SpawnsFromMeteorites>true</SpawnsFromMeteorites>
    <MaterialTypeName>Thorite</MaterialTypeName>
    <MinedOre>ThoriteOre</MinedOre>
    <MinedOreRatio>5</MinedOreRatio>
    <CanBeHarvested>true</CanBeHarvested>
    <IsRare>true</IsRare>

<AsteroidGeneratorSpawnProbabilityMultiplier>2</AsteroidGeneratorSpawnProbabilityMultiplier>
    <IsIndestructible>false</IsIndestructible>
    <DamageRatio>0.2</DamageRatio>
    <ParticleEffect>Collision_Meteor</ParticleEffect>

        <VoxelHandPreview>Textures\Voxels\Thorite_thumbnail.dds</VoxelHandPreview>

        <ColorMetalXZnY>Textures\Voxels\Thorite_AxisXZ_cm.dds</ColorMetalXZnY>
                        <ColorMetalY>Textures\Voxels\Thorite_AxisY_cm.dds</ColorMetalY>
                        <ExtXZnY>Textures\Voxels\Thorite_AxisXZ_add.dds</ExtXZnY>

        <NormalGlossXZnY>Textures\Voxels\Thorite_AxisXZ_ng.dds</NormalGlossXZnY>
                        <NormalGlossY>Textures\Voxels\Thorite_AxisXZ_ng.dds</NormalGlossY>

                        <ColorMetalXZnYFar1>Textures\Voxels\Thorite_AxisXZ_distance_cm.dds</
ColorMetalXZnYFar1>
                        <ExtXZnYFar1>Textures\Voxels\Thorite_AxisXZ_add.dds</ExtXZnYFar1>

        <NormalGlossXZnYFar1>Textures\Voxels\Thorite_AxisXZ_ng.dds</NormalGlossXZnYFar1>
    <Friction>1</Friction>
    <Restitution>1</Restitution>
   </VoxelMaterial>
  </VoxelMaterials>
</Definitions>
```

Lets talk about this a bit, I do not know what all of the values are exactly, but those I do I will cover here.

<VoxelMaterials> this tag tells the game engine, that data for voxels will reside here

<VoxelMaterial xsi:type="MyObjectBuilder_Dx11VoxelMaterialDefinition"> this tag tells the engine, what follows is one of the voxel defintions, you can have many

<SubtypeId>Thorite</SubtypeId> will always be the name of your ore.

<SpawnsInAsteroids>true</SpawnsInAsteroids> this will make this ore spawn in the asteroid field

<SpawnsInPlanets>true</SpawnsInPlanets> You need to set this to spawn on planets, but you will need to work with an additional file to get it to work

<SpawnsFromMeteorites>true</SpawnsFromMeteorites> Gives the ore a chance to be in a meteorite

<MinedOre>ThoriteOre</MinedOre> will always be the name of your ore, We create this ore in the PhysicalItems.sc later

<MinedOreRatio>5</MinedOreRatio> How much ore you get by mining, a lower value means you take longer to mine the ore in smaller quantities, while a higher number means the opposite.

<CanBeHarvested>true</CanBeHarvested> If you want to harvest it, this needs to be true.
<IsRare>true</IsRare> I have found, if you set rare to false, it does not show up anywhere, so keep this true

<AsteroidGeneratorSpawnProbabilityMultiplier>2</AsteroidGeneratorSpawnProbabilityMultiplier> How likely it is to spawn, this is 2x

The other values I am not sure of, but they are kept here for posterity, also notice, the sections that contain the filepaths to the images we created
are reused a few times, so this leaves this open for you to get even more creative with your textures, as this is the least amount of files
I could use to make this work, other examples may show more tags that can be used with images, but for now this is enough.

We now move on to PhysicalItems.sbc this file describes the items that can exist in the game, and the image to use
we will have both the ore, and the ingot created from the ore that can be dropped as a 3d object in game.
We will call our file Tutorial_PhysicalItems.sbc

It looks like this.

```
<?xml version="1.0"?>
<Definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <PhysicalItems>
          <PhysicalItem>
           <Id>
       <TypeId>Ore</TypeId>
       <SubtypeId>ThoriteOre</SubtypeId>
     </Id>
     <DisplayName>Thorite Ore</DisplayName>
     <Icon>Textures\GUI\Icons\Ore\Thorite_ore.dds</Icon>
     <Size>
       <X>0.07</X>
       <Y>0.07</Y>
       <Z>0.07</Z>
     </Size>
     <Mass>0.8</Mass>
     <Volume>1.8</Volume>
     <Model>Models\Components\Sphere.mwm</Model>
     <IconSymbol>Tr</IconSymbol>
   </PhysicalItem>
 <PhysicalItem>
     <Id>
       <TypeId>Ingot</TypeId>
       <SubtypeId>ThoriteIngot</SubtypeId>
     </Id>
     <DisplayName>Thorite Ingot</DisplayName>
     <Icon>Textures\GUI\Icons\Ingot\Thorite_ingot.dds</Icon>
     <Size>
       <X>0.072</X>
       <Y>0.072</Y>
       <Z>0.072</Z>
     </Size>
     <Mass>0.8</Mass>
     <Volume>1.8</Volume>
     <Model>Models\Ingots\iron_ingot.mwm</Model>
     <PhysicalMaterial>Metal</PhysicalMaterial>
     <IconSymbol>Tr</IconSymbol>
   </PhysicalItem>
  </PhysicalItems>
</Definitions>
```

<PhysicalItems> tells the game engine we have a list of physical items next

<PhysicalItem> the item we want to describe.

<TypeId>Ore</TypeId> we are using 2 type ids for this, one for the Ore, one for the Ingot, we describe one after the other in full

<SubtypeId>ThoriteOre</SubtypeId> This is the ID we use to refer to this ore, used above in VoxelMaterials

<DisplayName>Thorite Ore</DisplayName> The name you see in game when it is floating in space, or laying on the ground

<Icon>Textures\GUI\Icons\Ore\Thorite_ore.dds</Icon> This is where we saved our icons

<Size> We have X,Y,Z here, this will decribe how large the 3d objects are.

<Mass>0.8</Mass> How heavy it is

<Volume>1.8</Volume> How much space it takes up

<Model>Models\Components\Sphere.mwm</Model> We just use the default model here, a round sphere

<IconSymbol>Tr</IconSymbol> The abbreviation for this elememt, also seen in game by ore detector

We add two entries here, one to describe the ore, another to describe the bar, and that is all we needs

Now we move to the Blueprints.sbc file, ours will be named Tutorial_Blueprints.sbc

```
<?xml version="1.0"?>
<Definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Blueprints>
    <Blueprint>
      <Id>
        <TypeId>BlueprintDefinition</TypeId>
        <SubtypeId>ThoriteToIngot</SubtypeId>
      </Id>
      <DisplayName>Thorite Ingot</DisplayName>
      <Icon>Textures\GUI\Icons\ingot\Thorite_ingot.dds</Icon>
      <Prerequisites>
        <Item Amount="1" TypeId="Ore" SubtypeId="ThoriteOre" />
      </Prerequisites>
      <Result Amount="0.5" TypeId="Ingot" SubtypeId="ThoriteIngot" />
      <BaseProductionTimeInSeconds>2</BaseProductionTimeInSeconds>
    </Blueprint>
  </Blueprints>
</Definitions>
```

<Blueprints> Tells the game engine we have a list of blueprint items coming

<Blueprint> The blueprint we want to describe

<SubtypeId>ThoriteToIngot</SubtypeId> This is the blueprint name, we will use it later, it is just our material with ToIngot added

<Icon>Textures\GUI\Icons\ingot\Thorite_ingot.dds</Icon> The ingot image we created

```
      <Prerequisites>
        <Item Amount="1" TypeId="Ore" SubtypeId="ThoriteOre" />
      </Prerequisites>
```

The Prerequisites section, is what is required to create this item, and how much of it we need to do so. For now we just want to turn 1 Ore into 1 Ingot, the SubtypeId will be the SubtypeID of the Ore we created in PhysicalItems

<Result Amount="0.5" TypeId="Ingot" SubtypeId="ThoriteIngot" /> this is what we get after smelting this ore and it points to the physical item we made for the ingot

TypeId="Ingot" is important here, this means a regular refinery will smelt this item into a bar
Other choices are CommonMetals for basic refinery and SurvivalKitIngots for survival kits

<BaseProductionTimeInSeconds>2</BaseProductionTimeInSeconds> How many second this takes to smelt per bar

And that is all we need for Tutorial_Blueprints.sbc

Next is BluePrintClasses.sbc we will call ours Tutorial_BluePrintClasses.sbc

```
<?xml version="1.0"?>
<Definitions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
	<BlueprintClassEntries>
	<Entry Class="Ingots" BlueprintSubtypeId="ThoriteToIngot" />
	</BlueprintClassEntries>
</Definitions>
```

<BlueprintClassEntries> Tells the engine we have blueprint entries

<Entry Class="Ingots" BlueprintSubtypeId="ThoriteToIngot" /> we are adding the ingot blueprint we created to the get

And thats it

Now we go back and select our MyMod directory, copy it to
C:/Users/<username>/AppData/Roaming/SpaceEngineers/Mods
and rename it Tutorial
Now start the game, create a new creative game of any senario, add this new mod, which you should see on the top left of the list, and we will test the voxel with voxel hands