Asteroid Mining

Team name: Pied Pipers

Supervisor:

Dr. Balla Katalin

Members:

Abdelrahman Desoki	DOHDZF	adesoki@edu.bme.hu
Neda Radonjic	NQC17Y	neda.radonjic@edu.bme.hu
Tushig Bat-Erdene	QBI3JH	bat-erdene.tushig@edu.bme.hu
Chaitanya Arora	BWTFX8	chaitanya.arora@edu.bme.hu
Janibyek Bolatkhan	BOI6FK	janibyekbolatkhan@edu.bme.hu
Kasay Ito	LX5RFB	kasay.ito@edu.bme.hu

March 6, 2022

3. Analysis model – version 1

3.1 Object catalog

3.1.1 Settler

The settler is the main protagonist of the game with a goal to build the space station from the resources of the asteroids. It can perform various tasks by itself like traveling in the spaceship through the asteroid belt, wandering in the asteroid belt to find resources, drilling, mining, building robots, building teleportation-gates and using them for transport, and hiding in the asteroid during danger situations. Sun Storm and radioactive explosions can affect the settlers. Responsibilities of the settler involve finding adequate resources to build the space station and collecting them on one asteroid in order to win the game, keep himself safe from the various dangers that he/she might encounter.

3.1.2 Spaceship

A one person spaceship is a vehicle used by each settler to travel between different asteroids in the asteroid belt. It is the main means of transport in the game. A spaceship has a capacity of 10 units of a resource.

3.1.3 Resources

Resources are a variety of minerals which can be found in the core of the asteroids. Settler must mine and collect the necessary resources to achieve his goal of building the space station. The resources can also be used to build robots and teleportation gates. Resources can be collected and transported by the settler, who can carry at most 10 units of a resource at a time. The important resources that can be found include water, ice, iron, carbon, uranium. However, some resources, such as uranium, are highly radioactive and can cause explosions of asteroids that they make up, which is a danger to settlers.

3.1.4 Teleportation gates

Apart from the spaceships, another transportation option are teleportation gates. Settlers can build them, provided that they have the needed resources which make up the gates - iron, water ice, and uranium. They come in pairs, and are deployed by the settlers, which can only carry two gates at a time. A settler/robot who enters the gate at one end will be teleported to the other end by the gate.

3.1.5. Robot

If they collect enough resources(iron, carbon, uranium), the settlers can build robots which are autonomous entities controlled by artificial intelligence. The robots can only travel/be teleported

between asteroids and drill. Since they are not able to transport things, they can not mine. Robots are also affected by dangers, but they can survive a radioactive explosion, only landing on a neighboring asteroid, and they can avoid damage from the sunstorm if they hide in a hollow asteroid.

3.1.6 Asteroid

In the asteroid belt, there are many asteroids between which the robots and settlers can travel. They are covered by rocks, and the depth of the rock mantle can vary from asteroid to asteroid.The asteroids either contain a single type of a resource/material (some of which are radioactive) in their core, or they are hollow. Settlers drill and mine the asteroids to extract resources, whereas robots can only drill them. If an asteroid is hollow, it can be filled by one unit of a resource by the settler. If an asteroid is fully drilled and has a radioactive core, it will explode when at perihelion, killing any settler on it. Hollow asteroids can serve as a shelter for settlers in case of a sun storm.

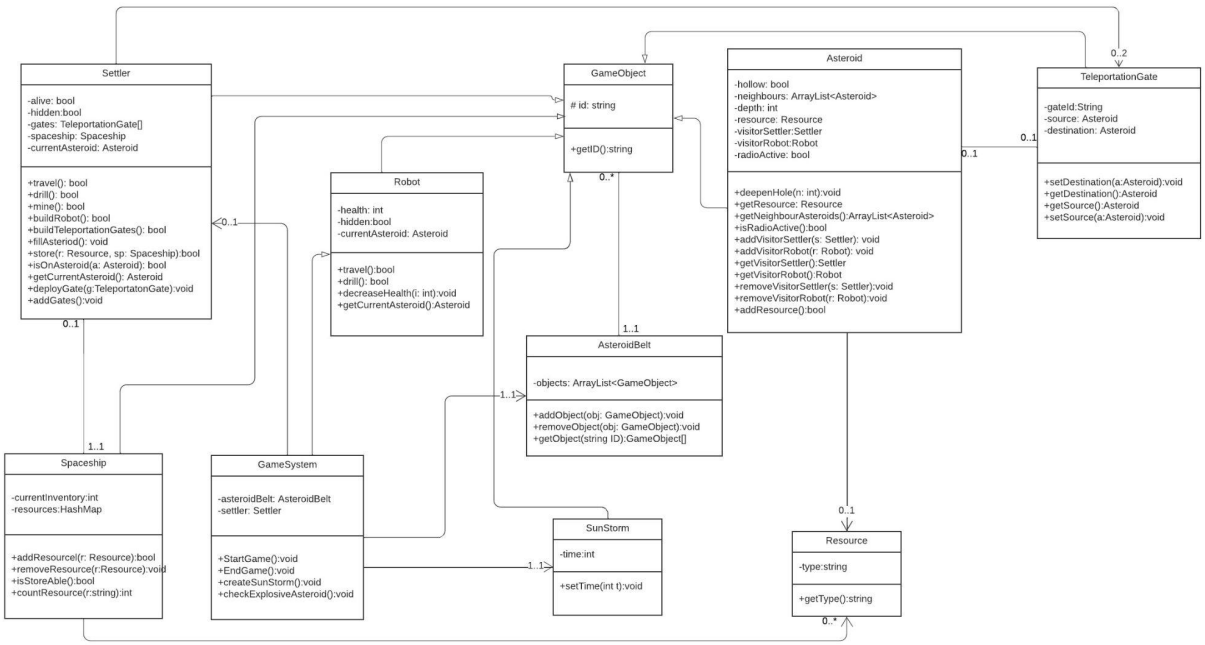
3.1.7 Space station:

The main objective of the game is for the settlers to collect enough resources to build a space station. The space station should consist of three units of each material, and can only be built when those resources are collected on one asteroid by the settlers.

3.1.8 Asteroid belt:

The asteroid belt is where all the asteroids are located and where all the events occur. It ‘hosts’ different elements of the game - the asteroids, the sun storms and explosions

3.2 Static structure diagrams



3.3 Class description

3.3.1 Settler

Responsibility

The settler travels between different asteroids in the asteroid belt, drills them, and mines the useful resources. The settler is responsible for collecting, transporting, and storing the resources on one asteroid (in order to win the game), as well as taking care of its safety by staying aware of the dangers that can affect it- sun storms and radioactive explosions. A settler can build AI-controlled robots, as well as teleportation gates- if it has enough resources. A settler has the goal of collecting 3 units of each resource on one asteroid, in order to win the game.

Attributes

- **bool alive:** specifies if the settler is alive or not at the moment, to ensure extendability, in case of requirements change (e.g. multiple settlers).
- **bool hidden:** specifies if the settler is hidden inside a hollow asteroid or not

- **TeleportationGate[] gates:** this attribute can have a value of maximum 2, because the settler can carry at most 2 gates (a pair) at a time.
- **Spaceship spaceship:** one settler knows exactly one spaceship
- **Asteroid currentAsteroid:** the asteroid where the settler stands on

Methods

- **bool travel()**- The settler attempts to travel to a neighboring asteroid. If the asteroid really is a neighbor and the operation is successful, the truth is returned.
- **bool drill()** - The settler drills through the mantle of an asteroid.
- **bool mine()** - If the mantle of the asteroid has been drilled through, the settler now mines the asteroid for useful resources.
- **bool buildRobot()** - We check if there are enough resources to build a robot. If so, a robot is instantiated, and true is returned.
- **bool buildTeleportationGates()** - We check if there are enough resources to build a teleportation gate. If so, a gate is instantiated, and a true is returned.
- **void fillAsteroid()** - The settler can fill a hollow asteroid with one unit of a resource.
- **bool hide()** - Checks if a settler is hidden in a hollow asteroid. If not, the action of hiding a settler is performed, and a true is returned.
- **bool store(Resource r, SpaceStation sp):** it stores the given resource to the space station. If there is no available free space it returns false. This method does not appear in the sequence diagram since the mine() method handles this functionality.
- **bool isOnAsteroid(Asteroid a):** return if the settler is on the asteroid or not.
- **Asteroid getCurrentAsteroid():** return the current Asteroid of Settler.
- **void deployGate(TeleportationGate g):** deploys the corresponding available teleportation gate.
- **void addGates(Teleportation g):** adds new gates to the list gates which Settler has right now.

Superclasses

- **GameObject**

3.3.2 Spaceship

Responsibility

A spaceship has the responsibility of transporting the settler and its collected resources through the asteroid belt. It is also used as a container of resources.

Attributes

- **int currentInventory:** stores the number of units of resources that the settler has stored in the spaceship already
- **HashMap resources:** describes the resources that the settler currently has. The key of the map is the type of the resource, and the value is the amount of the resource.

Methods

- **bool addResource(Resource r):** We attempt adding a unit of a resource into the spaceship. If there was enough capacity, and the unit was successfully added, we return true.
- **void removeResource(Resource r):** This method removes a unit of resource from the spaceship.
- **bool isStoreAble():** Checks if there is enough capacity in the spaceship to store another unit of resource.
- **int countResource(string r):** Counts the current amount of resources in the spaceship and returns an int value.

Superclasses

- **GameObject**

3.3.5 Robot

Responsibility

A robot can be built by a settler when enough resources are collected, and it is controlled by AI. It can drill, travel, and survive radioactive explosions. In the case of a sunstorm, it must hide inside a hollow asteroid, otherwise it will be damaged.

Attributes

- **int health:** measures the health of a robot. The robot can survive sun storms, but it will be damaged, so this attribute is needed to measure the health of the robot after a sun storm.
- **bool hidden:** determines whether the robot is hidden inside a hollow asteroid or not.
- **Asteroid currentAsteroid:** stores information about which asteroid the robot is currently on

Methods

- **bool travel():** when the robots attempt to travel to a neighboring asteroid, this method is called. If the robot successfully reaches the neighboring asteroid, true is returned.
- **void drill():** the robot can dril the rock mantle of the asteroid, to help the settler with resource extraction.
- **void decreaseHealth():** when the robot is not hidden during a sun storm, it gets damaged, and it health decreases
- **Asteroid getCurrentAsteroid:** gets information about the Asteroid on which the robot currently is.

Superclasses

- **GameObject**

3.3.4 Teleportation gates

Attributes

- **String gateId**: Stores the unique ID of a teleportation gate, to help us distinguish it.
- **Asteroid source**: Stores information about the Asteroid from which a robot or settler begins their journey through the gate.
- **Asteroid destination:** Stores information about the Asteroid to which a robot or settler will arrive through the gate.

Methods

- **void setDestination(Asteroid a):** Sets an Asteroid as a destination reached through the teleportation gate. Represents why this is necessary in a sequence diagram.
- **Asteroid getDestination():** Gets information about which Asteroid is the destination. Represents why this is necessary in a sequence diagram.
- **Asteroid getSource():** Sets an Asteroid as the source from which the traveler begins their journey through the gate. Represents why this is necessary in a sequence diagram.
- **void setSource(Asteroid a):** Gets information about the Asteroid from which the traveler begins their journey. Represents why this is necessary in a sequence diagram.

Superclasses

- **GameObject**

3.3.5 GameObject

Responsibility

GameObject is responsible for dealing with existing objects in the game. During the next analysis model or implementation, we will add the functions which are commonly used among derived classes.

Attributes

- **string id:** stores the unique ID of objects in the game

Methods

- **string getID:** gets the unique ID of an object

3.3.6 GameSystem

Responsibility

Only one GameSystem object can exist in a game. This class has the responsibility to start and end the game. These include condition checks, creating major entities and events.

Attributes

- **AsteroidBelt asteroidBelt:** This is an object of the class Asteroid belt this will; be used to create an asteroid belt in the game
- **Settler settler:** This is an object of the class settler which will later be used in the diagram.

Methods

- **void StartGame():** This function is used to initialize all major elements in the game.
- **void EndGame()**: this function is majorly invoked when the conditions which the settler can die become true.
- **void createSunStorm():** this function can be invoked randomly and creates an occurrence of a sunstorm in the game
- **void checkExplosiveAsteroid():** this function is called to check what all asteroids in the game are radioactive, and keep a track of them

3.3.7 AsteroidBelt

Responsibility

AsteroidBelt is a container that contains all necessary instances related to the played game. A settler, asteroids, robots and teleportation gates are contained.

Attributes

- **ArrayList<GameObject> objects:** all the list of the objects that currently exist on the asteroid belt, ranging from visitor to asteroid.

Methods

- **void addObject(GameObject obj):** Adds corresponding object to the **objects** list
- **void removeObject(GameObject obj):** removes corresponding object from the list.

- **GameObject[] getObject(String ID):** returns the list of objects related to the ID parameter.

3.3.8. Resource

Responsibility: The purpose of the resources class is to have a list of resources that are present in the game.

Attributes

- **string type**: This is used to store the name of the resource in a string attribute

Methods

- **getType():** This is a getter method used to get the resource name like iron, uranium etc.

3.3.9. Asteroid

Responsibility

An asteroid serves as a source of resources that our settlers are searching for, and they are of big importance during sun storms, because in the case when they are hollow, they serve as shelters for robots and settlers in danger.

Attributes

- **bool hollow:** Specifies whether the asteroid has a hollow core or not
- **ArrayList<Asteroid> neighbors:** Stores the information about what asteroids are neighbors of an asteroid.
- **int depth:** actual depth of the current asteroid but can be changed when it is being drilled
- **Resource resource:** a unit of resource this asteroid contains
- **Settler visitorSettler:** visitor Settler of the current asteroid, it is used for accessing the Settler visitor
- **Robot visitorRobot:** visitor Robot of the current asteroid, it is used for accessing the Robot visitor
- **bool radioActive:** determines if the asteroid is radioactive. Asteroid gets radioactive when it contains a resource- uranium.

Methods

- **void deepenHole(n: int):** when it is drilled, its depth will be decreased by n meters
- **Resource getResource:** returns a unit of resource which the current asteroid contains. We wrote this function and documentation because this is needed in the sequence diagram.
- **bool addResource():** adds a unit of resource the core of asteroid
- **ArrayList<Asteroid> getNeighbourAsteroids:** returns the list of neighboring asteroids of current asteroid
- **bool isRadioActive:** returns boolean value if the current asteroid is radioactive or not.
- **void: addVisitorSettler(s: Settler):** adds Settler visitor to the asteroid
- **void addVisitorRobot(r: Robot):** add Robot visitor to the asteroid
- **void removeVisitorSettler():** removes the settler that was visiting the asteroid
- **void removeVisitorRobot():** removes the robot that was visiting the asteroid
- **Settler getVisitorSettler:** gets information about the settler that is currently on the asteroid
- **Robot getVisitorRobot:** gets information about the robot that is currently on the asteroid

SuperClass

- **GameObject**

3.3.10. SunStorm

Responsibility

The reason why we have SunStorm class: because sunstorm exists as a real object in the game: the user can notice the existed sunstorm

Attributes

- **int time:** The time period over which a SunStrom happens-how long it lasts

Methods

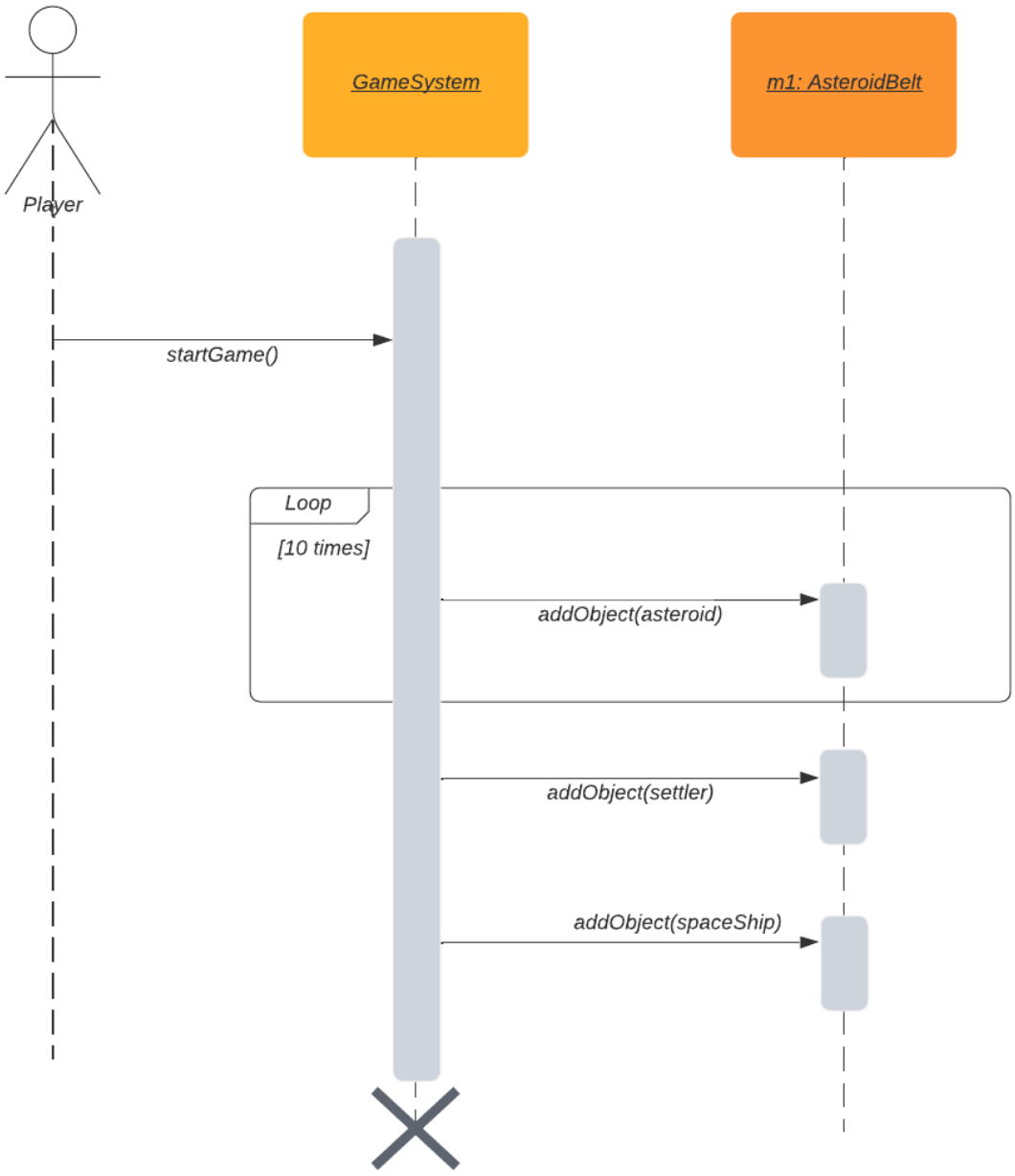
- **void setTime(int t)**: Sets the time period for the SunStorm.

SuperClass

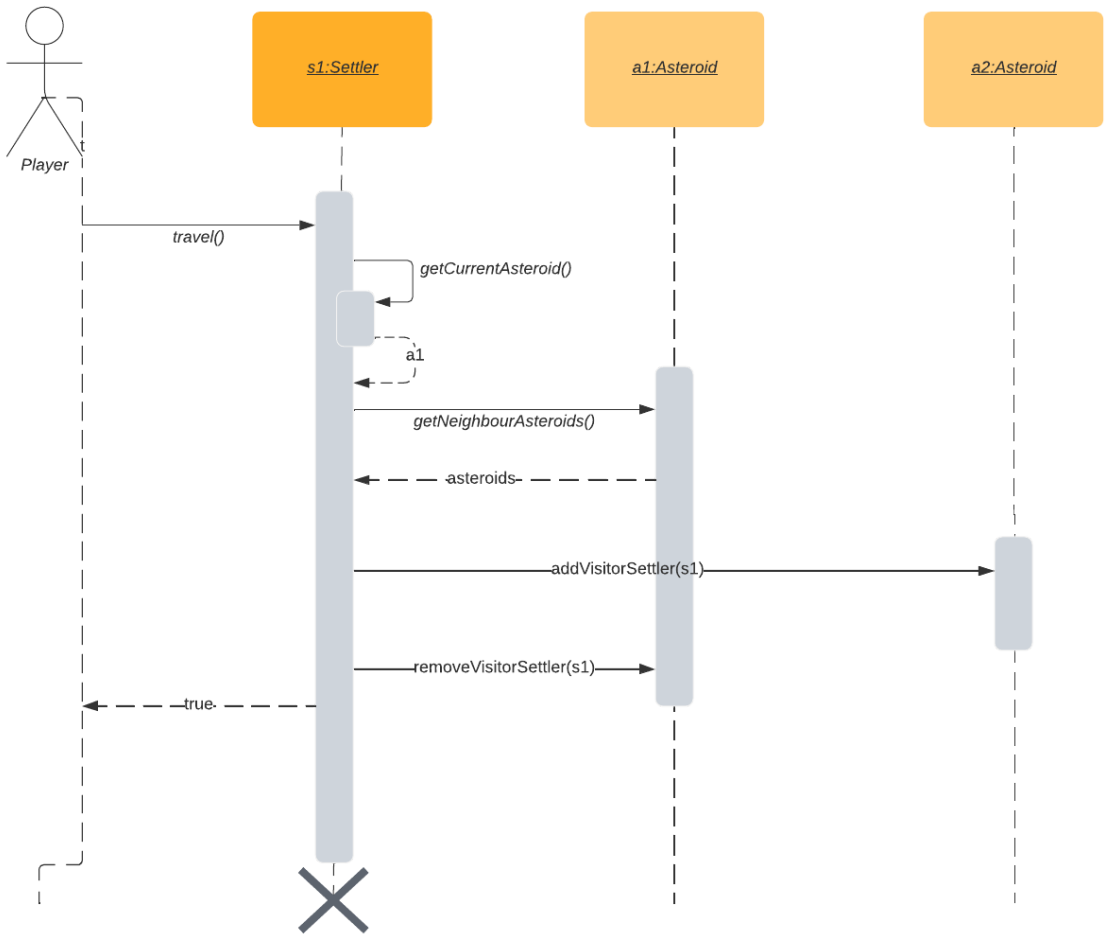
- **GameObject**

3.4 Sequence diagrams

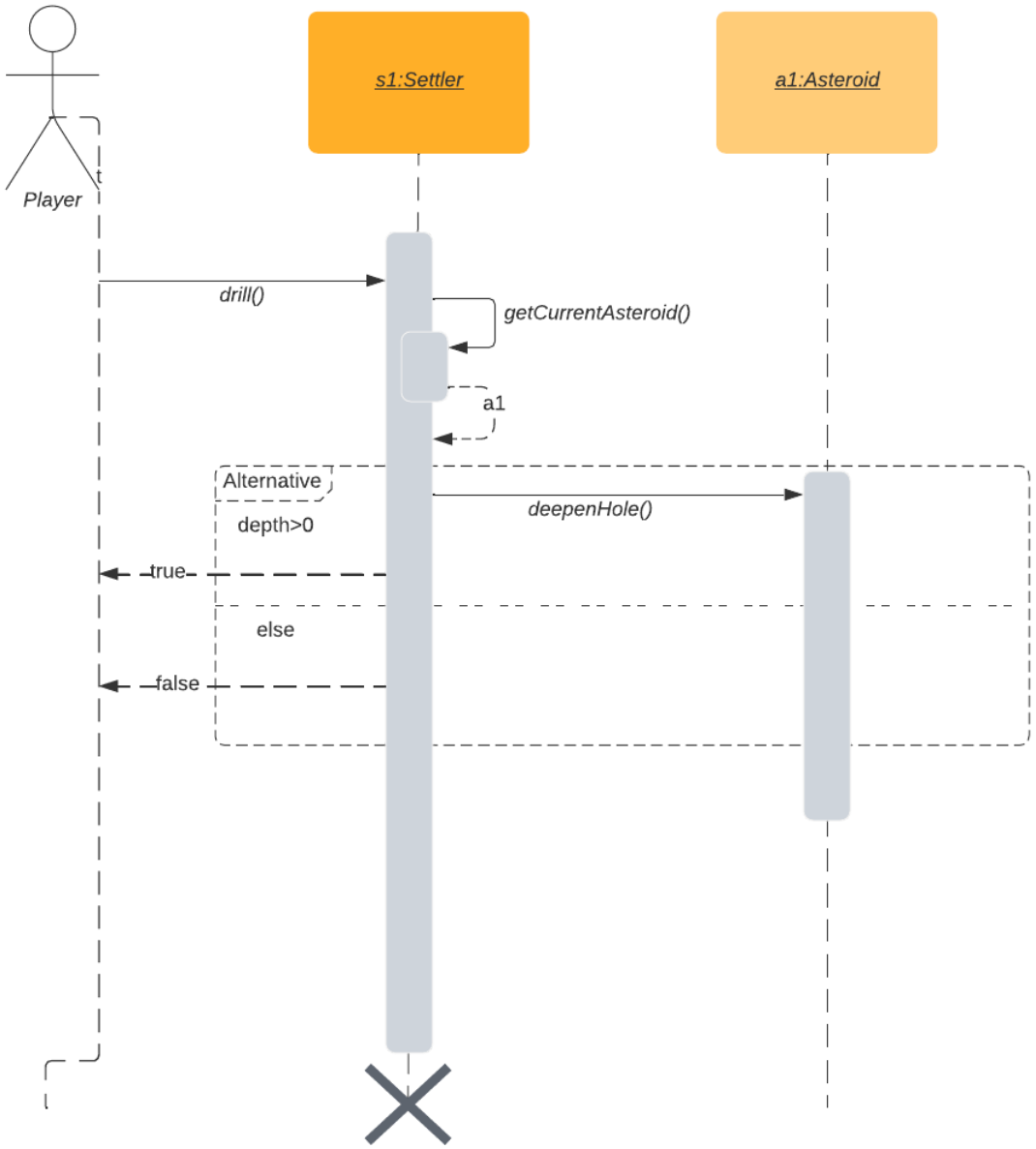
Player begins Game:



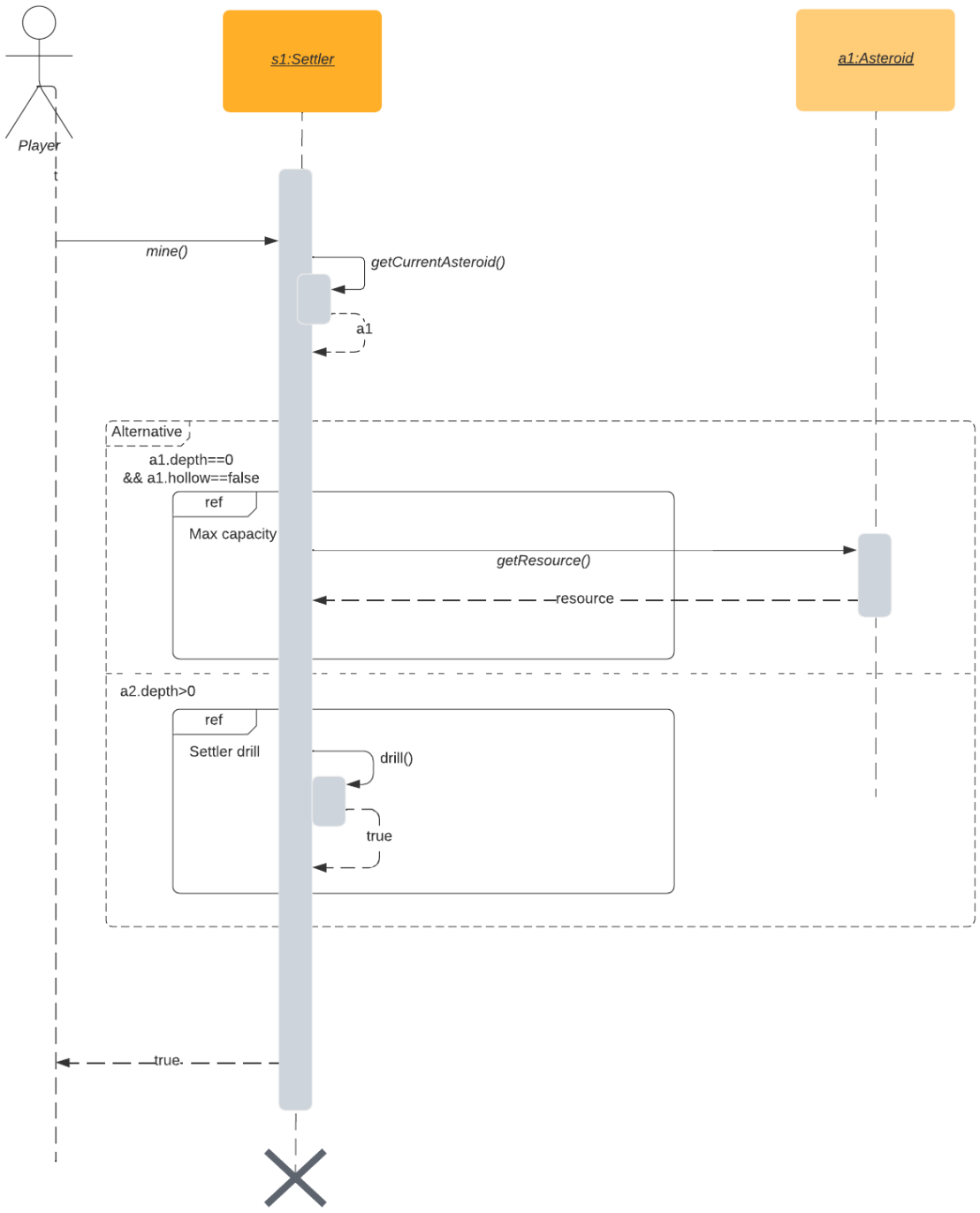
Settler travels:



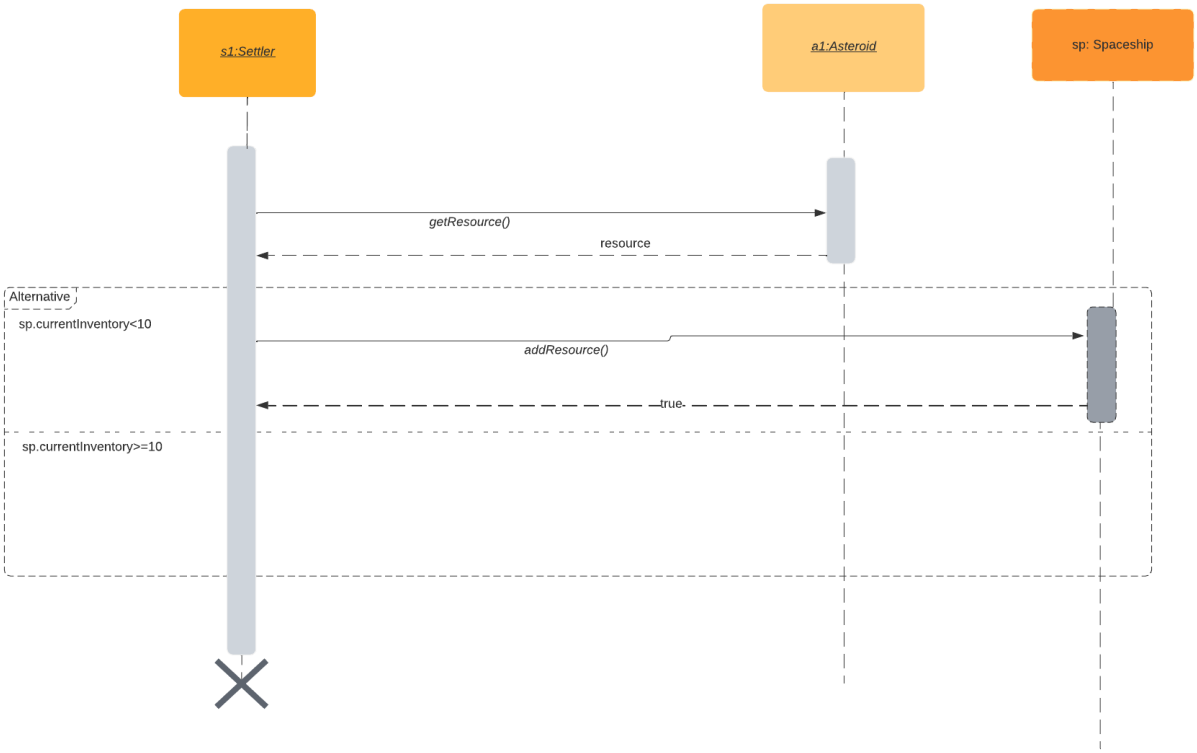
Settler Drills:



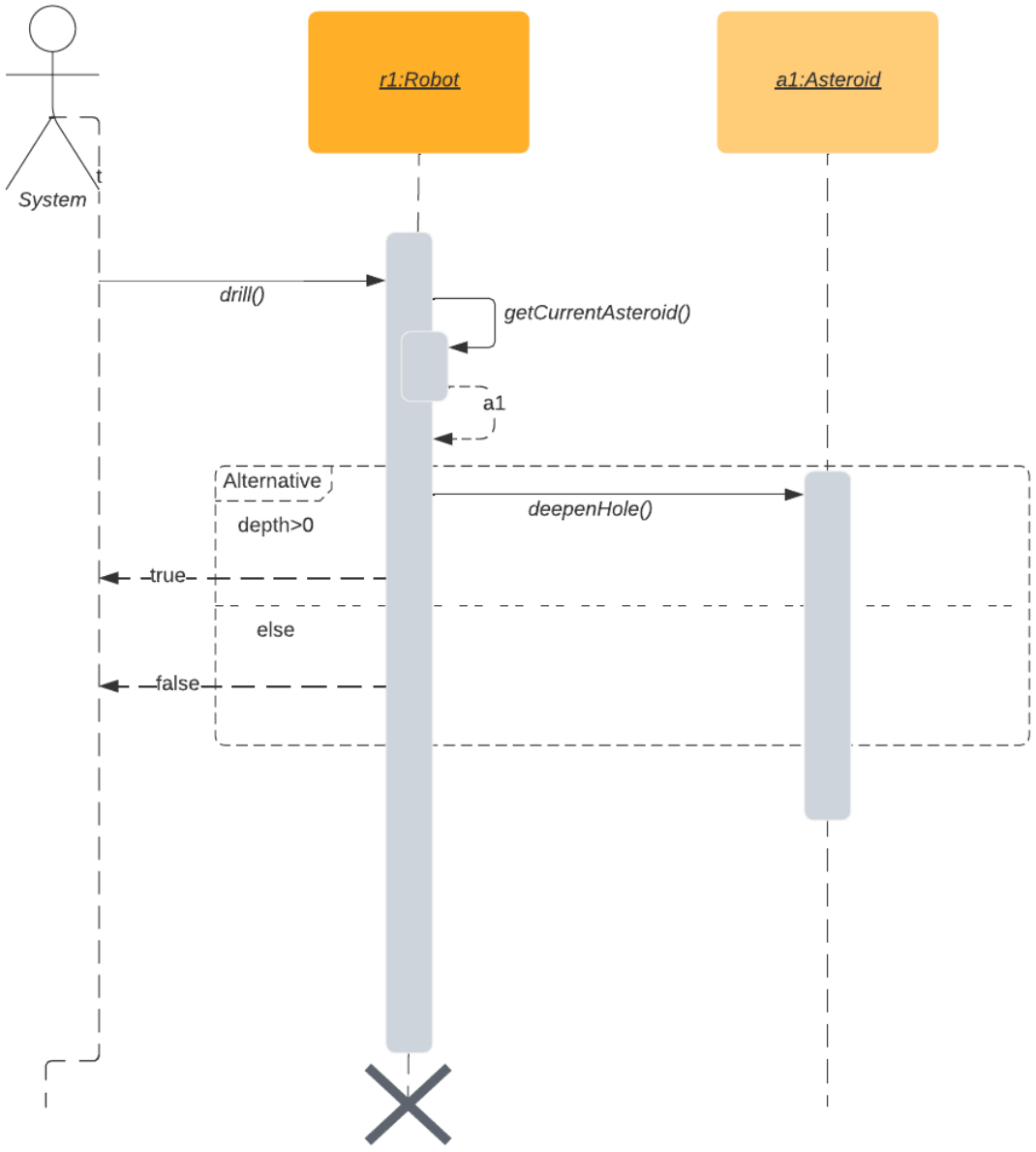
Settler Mines:



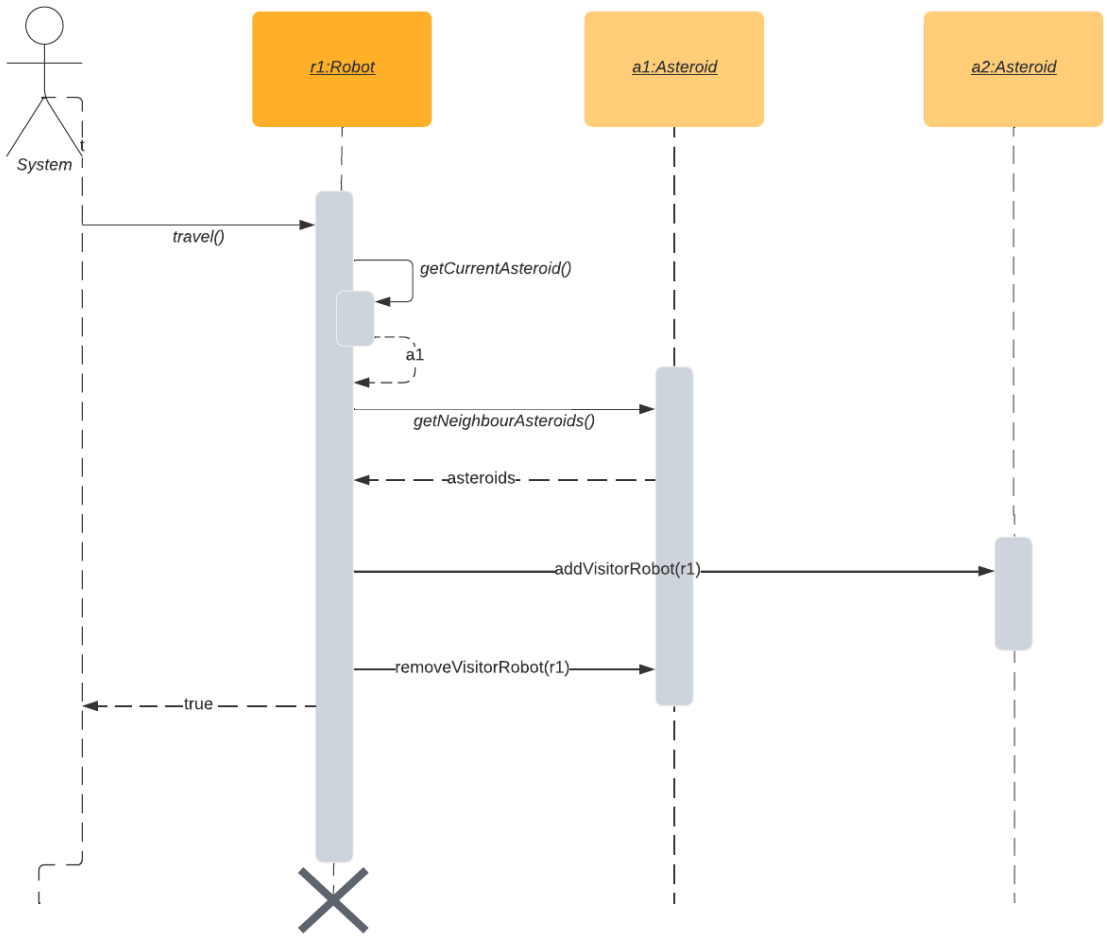
Max Capacity:



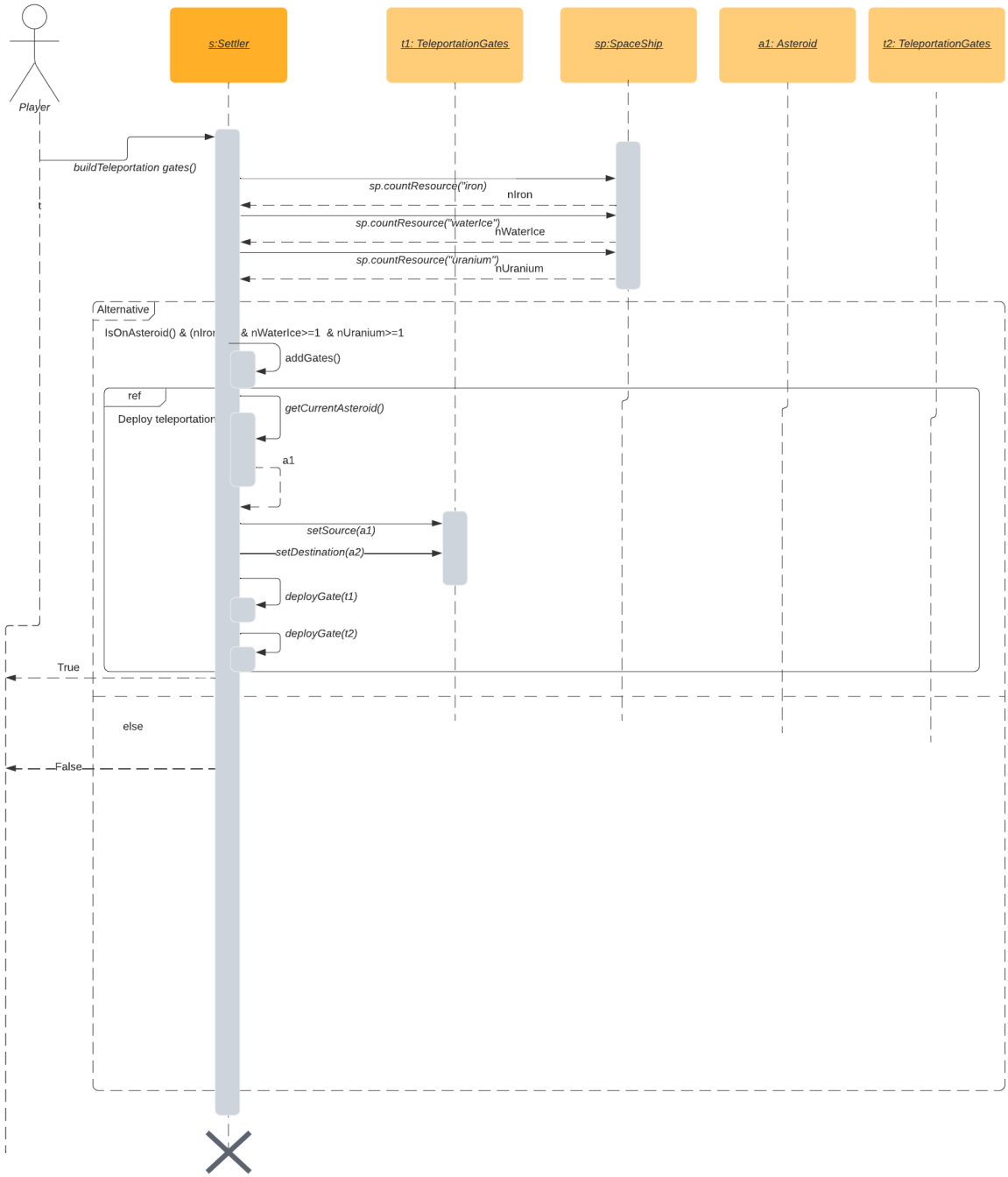
Robot drills:



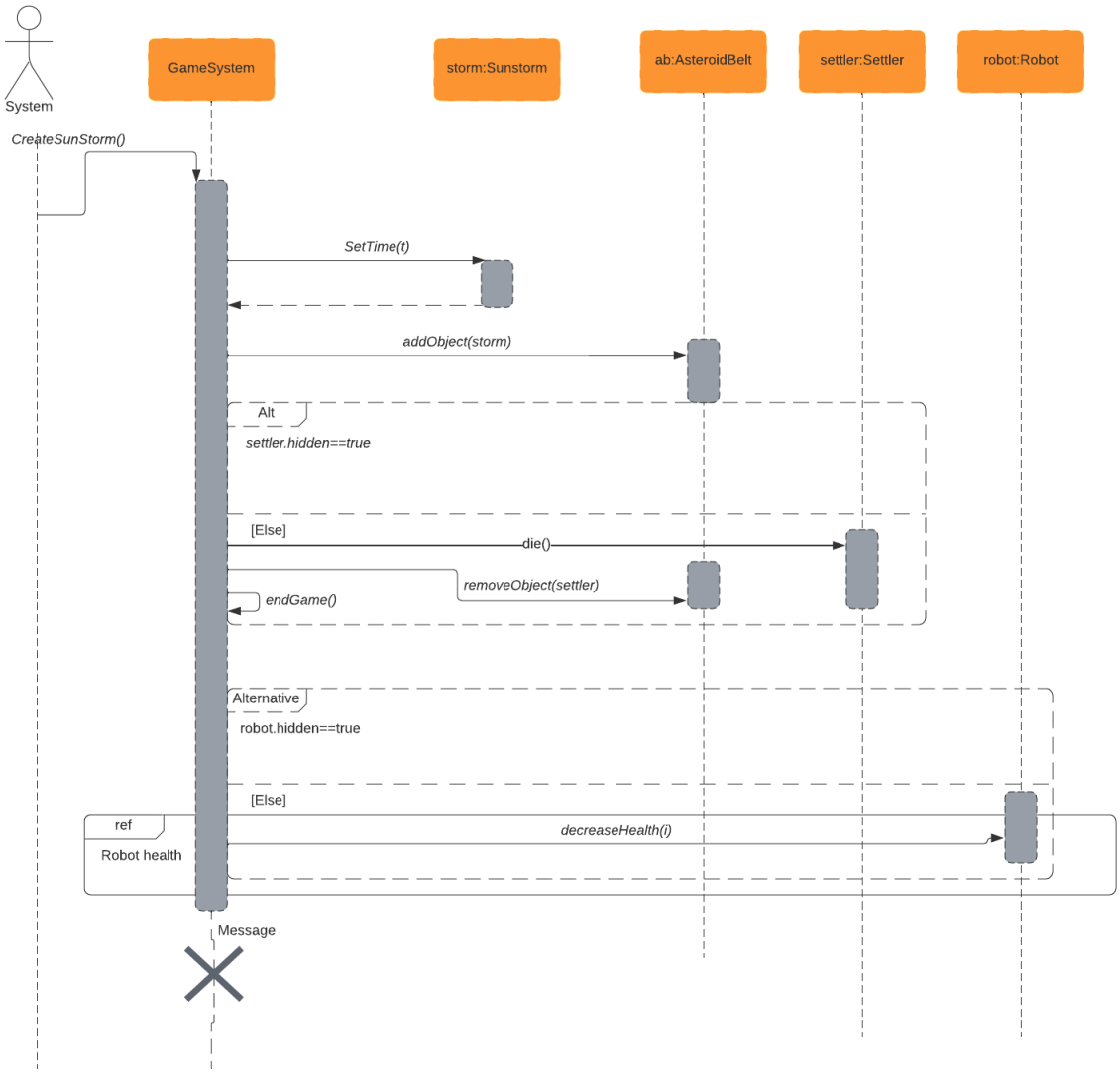
Robot travels:



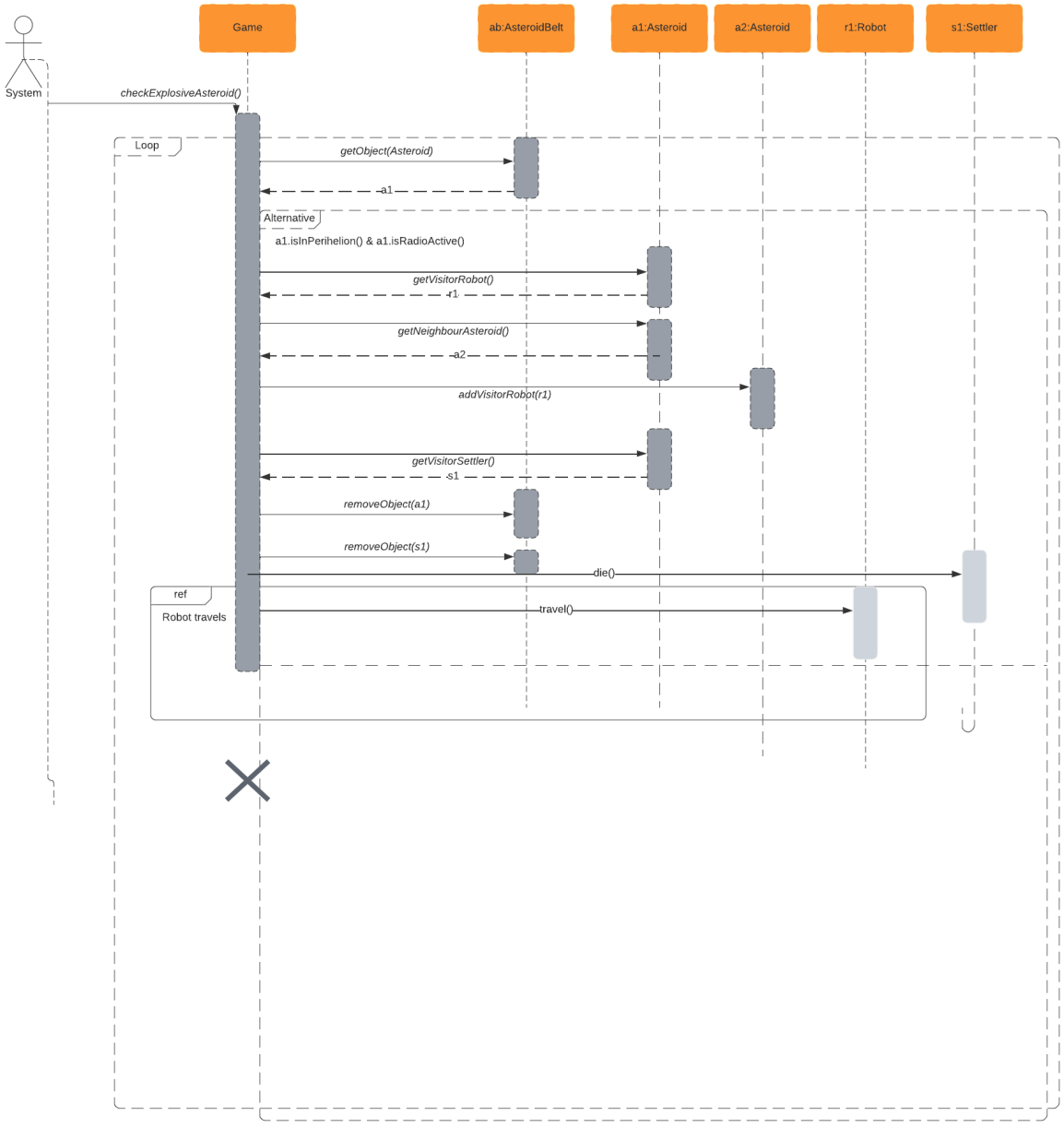
Build Teleportation-gate:



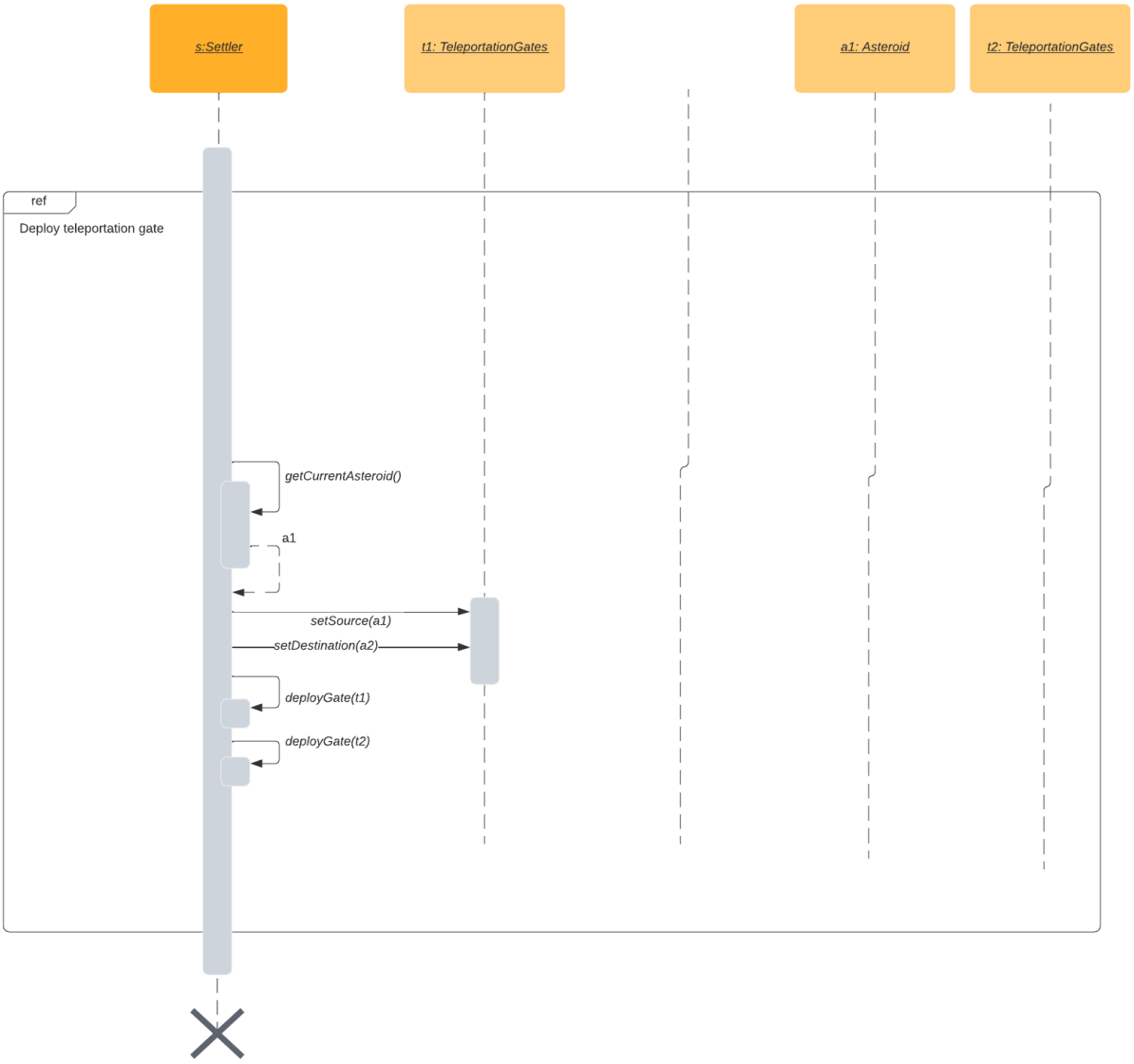
Sun Storm:


System
GameSystem
storm:Sunstorm
ab:AsteroidBelt
settler:Settler
robot:Robot
CreateSunStorm()
SetTime(t)
addObject(storm)
Alt
settler.hidden==true
[Else]
die()
removeObject(settler)
endGame()
Alternative
robot.hidden==true
[Else]
ref
Robot health
decreaseHealth(i)
Message

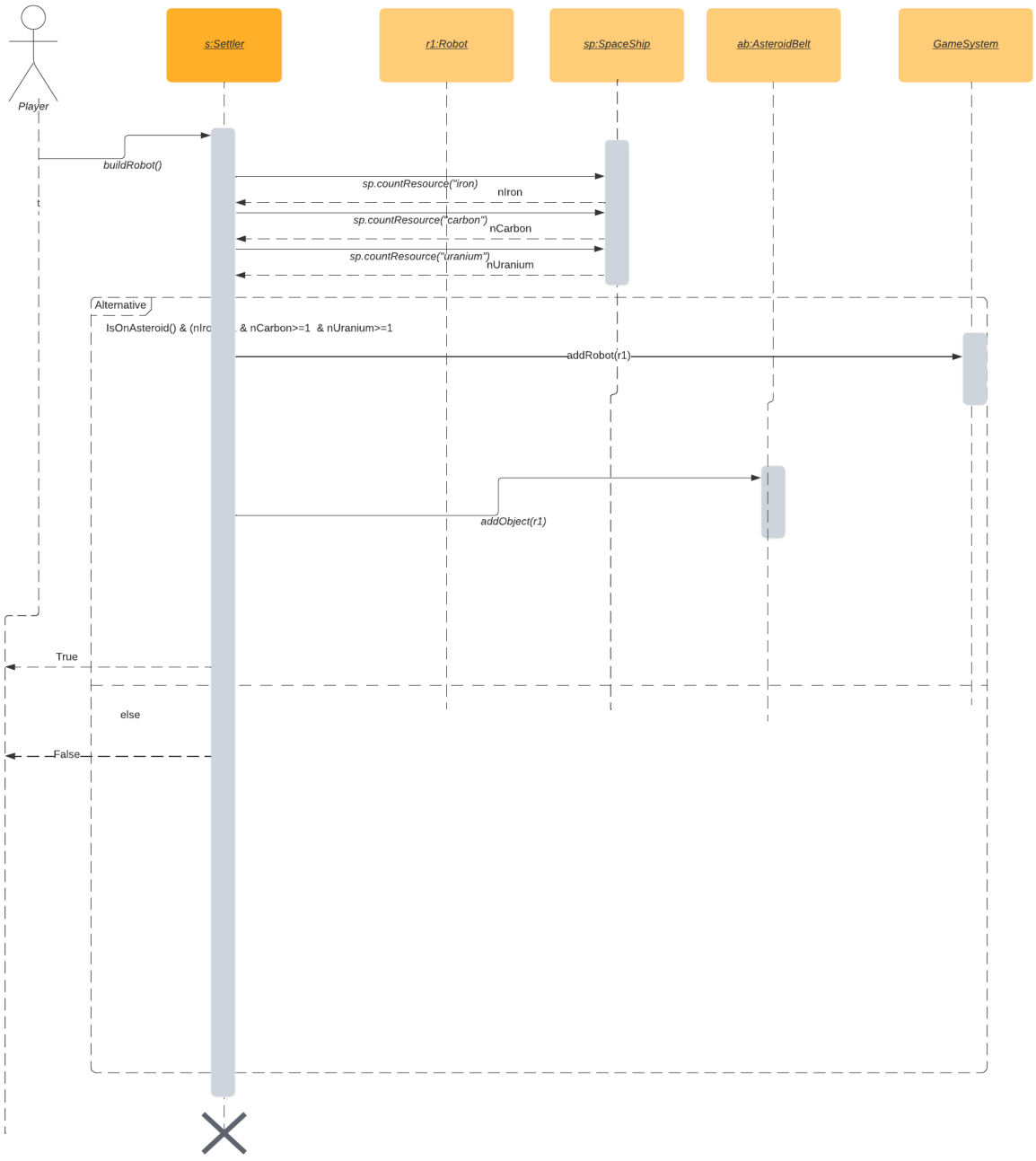
Asteroid Explodes:



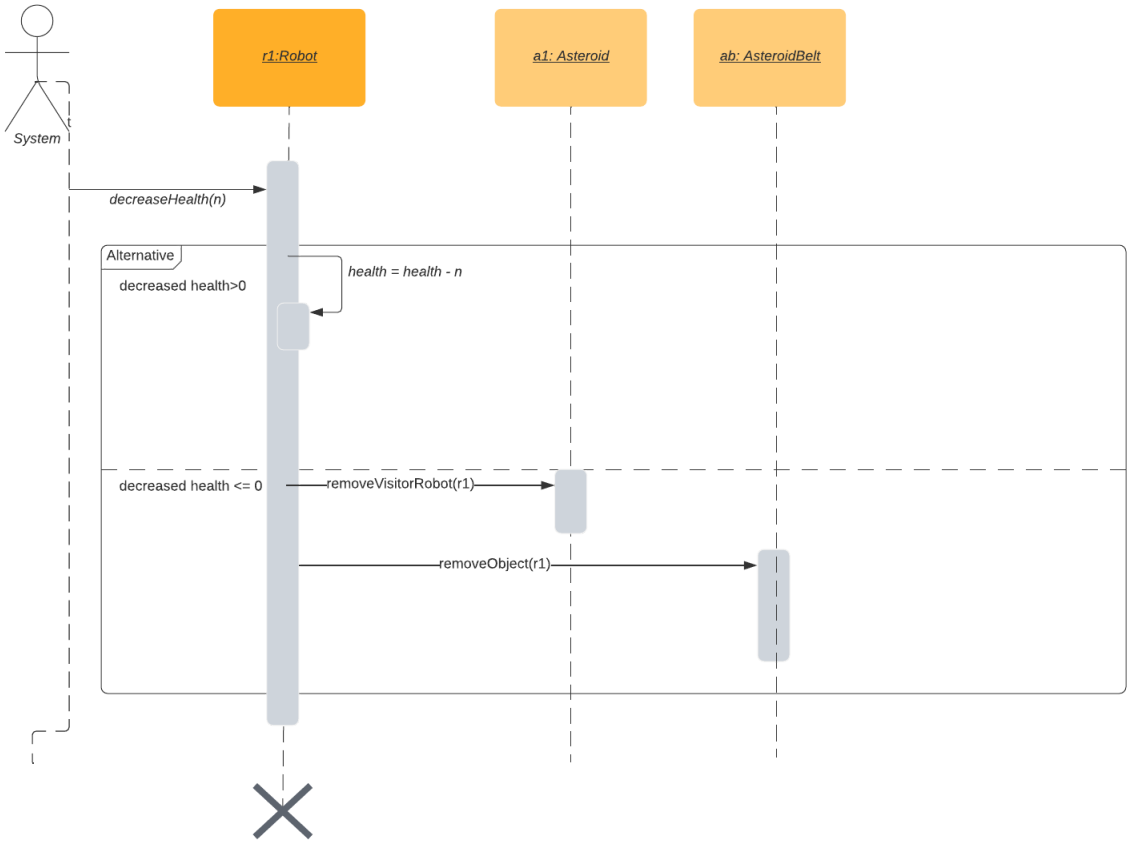
Deploy teleportation gate:



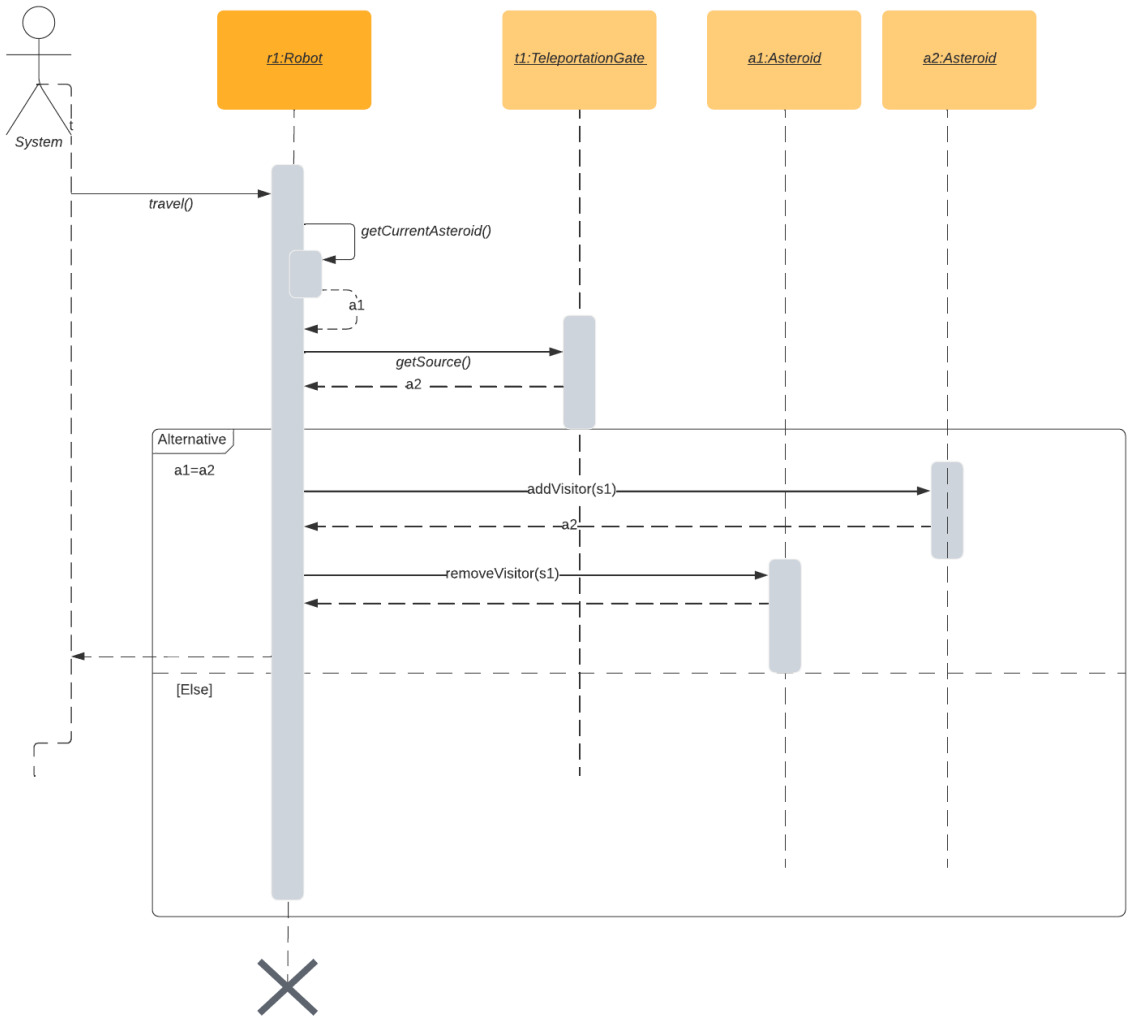
Build Robot:


Player
s:Settler
r1:Robot
sp:SpaceShip
ab:AsteroidBelt
GameSystem
buildRobot()
sp.countResource("iron")
nIron
sp.countResource("carbon")
nCarbon
sp.countResource("uranium")
nUranium
Alternative
IsOnAsteroid() & (nIron & nCarbon>=1 & nUranium>=1
addRobot(r1)
addObject(r1)
True
else
False

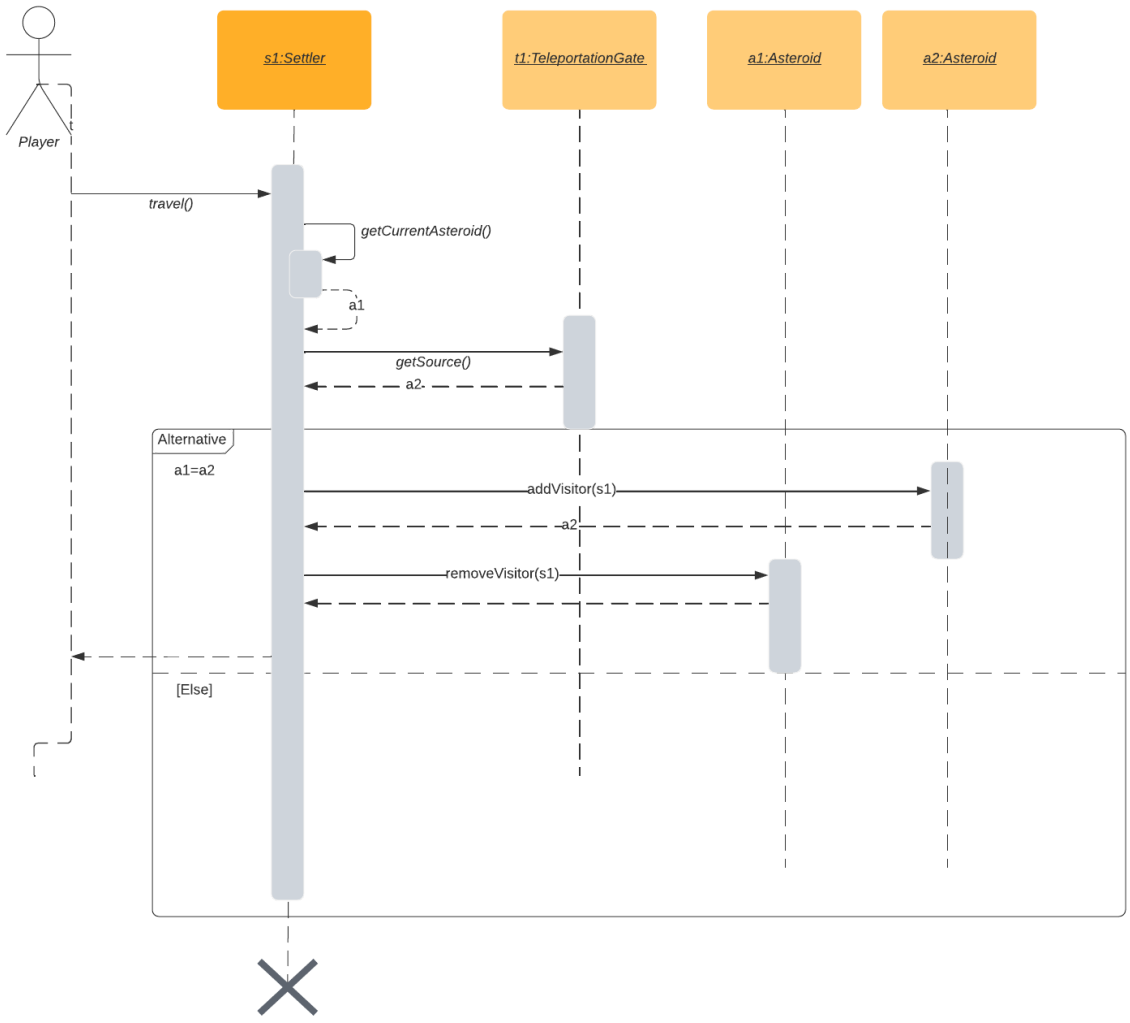
Robot health:



Robot passes through teleportation-gate:

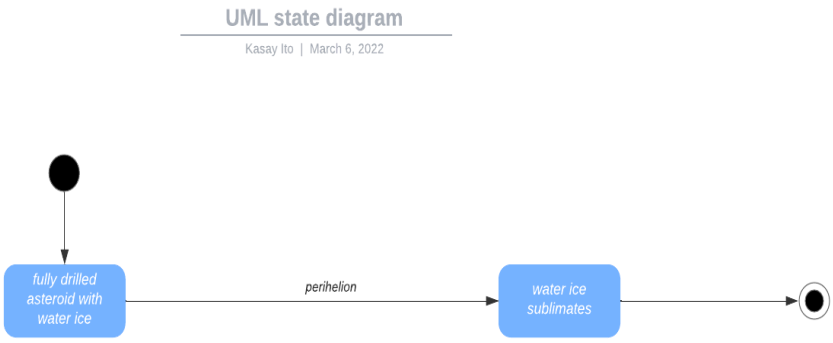


Settler passes through teleportation-gate:

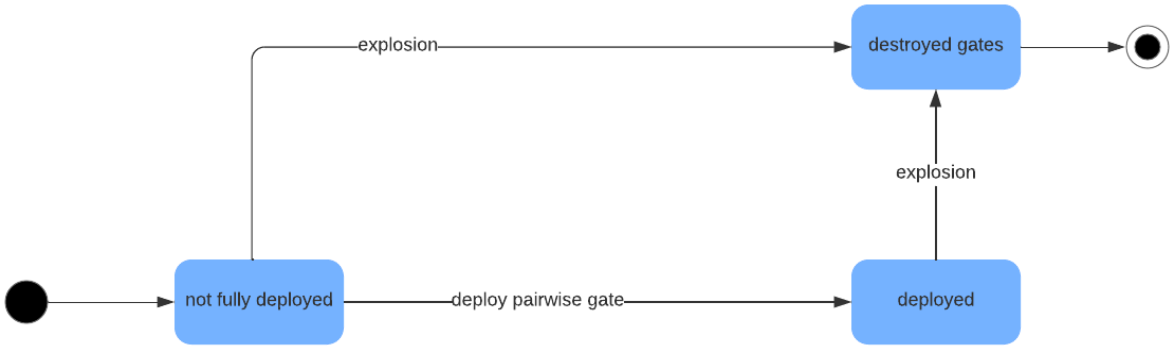


3.5 State-charts

1. Ice sublimates in asteroid

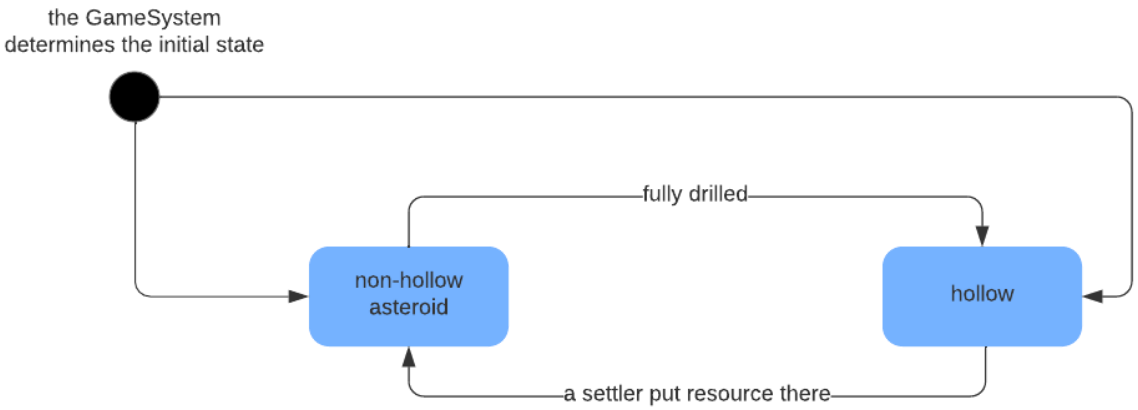


2. Teleportation gates deployment

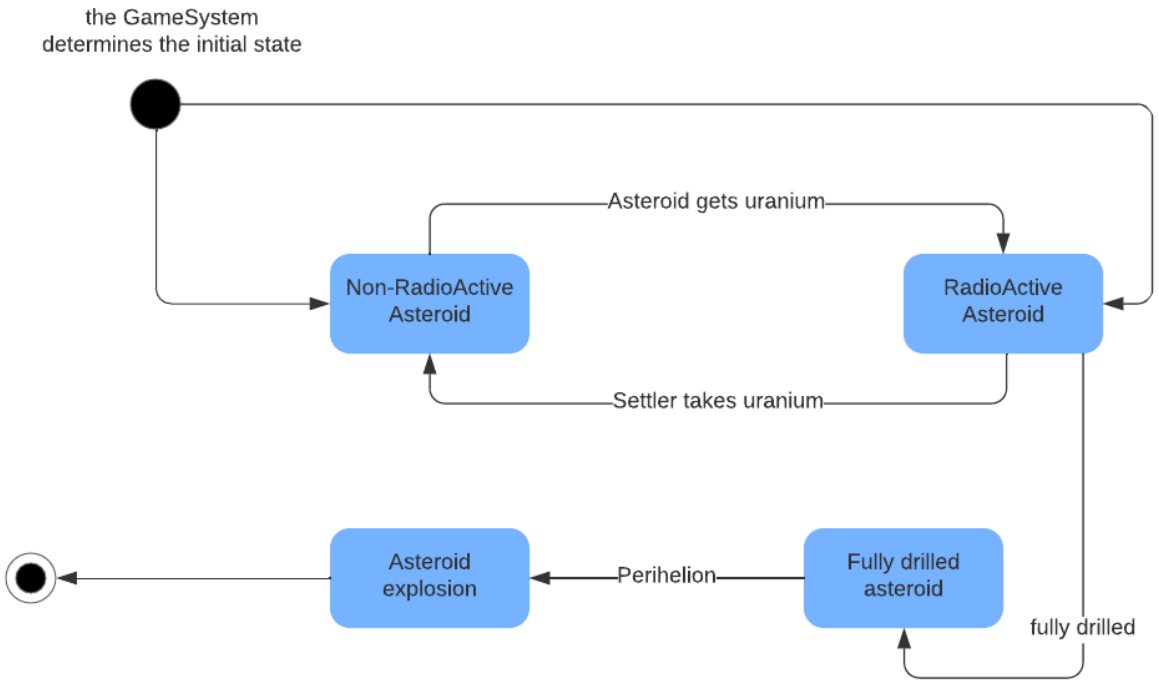


3. Settler lives

4. Hollow Asteroid

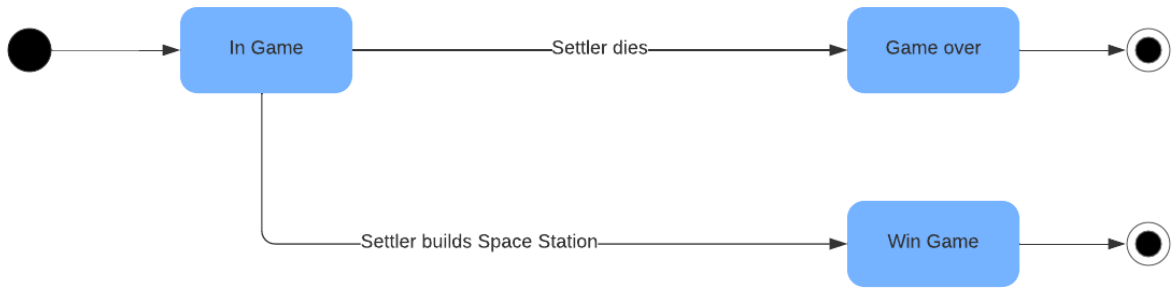


5. Asteroid explodes



6. Robot dies

7. Game ends



3.6 Protocol

Start (date & time)	Duration (hours)	Performer(s) name	Activity description

27/02/2022 Meeting in person	2,5 hours	All members	Meeting Reflecting on the mentor's comments for the requirements specification and starting to work on the analysis model. We identified the most important entities and what they will be responsible for. We started working on the class diagram.
01/03/2022 Online meeting- google meets	3 hours	All team members	Continuing to work on the class diagram, discussing the needed methods, and their naming, using the Lucid Chart tool.
02/03/2022 In person meeting	3 hours	All team members	Wrapping up the work on the first version of the class diagram, and specifically discussing and planning out GameSystem, Map and SunStorm classes. Discussing how we are visualizing the game at the moment - the number of asteroids, for example. Starting work on the sequence diagrams, as we have come to a common conclusion that this will take us the most time.
03/03/2022	4 hours	All team members	Continuing work on the sequence diagrams. Realized that due to schedule issues, we will have to split some of the

Meeting in person			tasks between ourselves and work on them in smaller groups or individually.
04/03/2022 Meeting in person	3 hours	All team members	Continuing the work, with tasks split between ourselves. Desoki and Chaitanya worked on the state diagrams, Janibyek and Kasay completed the remaining sequence diagrams, and Neda worked on the documentation, with others' inputs and help.
04/03/2022 Meeting in person	3.5 hours	Kasay ito alone	Sequence diagrams are refined and checked with the class diagram.
06/03/2022 Online meeting - Teams	3 hours	All team members	Completed State Chart diagrams and worked on the remaining parts of documentation. We synchronized the class diagram, sequence diagram and state machine diagram.