# 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 15, 2022**

# 4. Analysis model – version 2

## 4.1 Object catalog

### 4.1.1. Visitor

A visitor is either a robot or a settler, and depending on this, it has different responsibilities and capabilities. What is common to all visitors is that they travel between different asteroids in the asteroid belt, looking for resources. Making use of the OOPs concepts. We used inheritance.Settlers, they are capable of more-they can mine, unlike robots, and they can build robots and teleportation gates. Robots, once they are built by settlers, can perform the action of drilling the asteroids. Settlers are more affected by dangers - sun storms, and radioactive explosions, whereas robots survive explosions. Their responsibilities are highlighted more accurately in individual object descriptions and common ones are represented in the visitor class.

### 4.1.2. Settler

The settler is the main protagonist of the game with the goal of collecting enough resources from asteroids to build a space station. 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.

### 4.1.3. Robot

If they collect one unit of each iron, carbon, and 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. They are responsible for drilling, and also hiding when sandstorms occur, in order to stay safe.

### 4.1.4. 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. Its responsibility is getting the visitors from one asteroid to another, and transporting their resources as well. This entity differs from the settler because a spaceship

exists graphically and the settler can get out of it and directly stand on the asteroid. This is needed when considering extensibility or requirement changes such as the spaceship has been affected by the sun storm or the settler can use the spaceship for hiding from the sun storm.

### 4.1.5. Resource

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.

### 4.1.6. Uranium

Uranium is one of the resources found in the core of asteroids. It differs from other resources because it is radioactive. Therefore, those asteroids that have a core made up from uranium can explode. Uranium is one of the resources responsible for making up the structure of robots and teleportation gates.

### 4.1.7. Water Ice

Water ice is one of the resources found in the core of asteroids. It differs from other resources, because when a fully drilled asteroid with a core made up from water ice is at perihelion, the water ice sublimates (disappears). Water ice is one of the resources responsible for making up the structure of teleportation gates.

### 4.1.8. Iron

Iron is one of the resources found in the core of asteroids. It is one of the resources responsible for making up the structure of robots and teleportation gates.

### 4.1.9. Carbon

Carbon is one of the resources found in the core of asteroids. It is one of the resources responsible for making up the structure of robots.

### 4.1.10. Place

Place has responsibility that it deals with common functionalities between Asteroid and Teleportation Gates. Therefore, the information related to neighboring places (asteroids or teleportation gates) and which visitor (settler or robot) is on there is described here, meaning that this entity is the way to know neighbors.

### 4.1.11. 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. The gates are responsible for teleporting their users (visitors) between neighboring asteroids between which they are deployed.

### 4.1.12. 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. Responsibilities of the asteroids are: being a source of useful resources, being shelters from danger, and hosting visitors.

### 4.1.13. Radioactive Asteroid

When the core of an asteroid consists of uranium, that asteroid is radioactive. Such structure of the core is responsible for explosions of asteroids, which can occur, killing settlers, and launching robots to different asteroids. The reason why this functionality does not depend on Resource is that in Object Oriented (OO) concept, the condition whether radio active or not is determined by state of the asteroid, according to the requirements which describes it as fully drilled and perihelion.

### 4.1.14. Sun Storm

The sun storm exists visually in the game and the user can notice that there is an sun storm. This entity has responsibility for existing so that the Game can compute the collision among other objects. If there is no such sun storm entity (i.e. class), then the user may not notice the event related to the sun storm and the Game cannot compute the collision between them. Consequently, this object is one of our assumptions.

### 4.1.15. Game

This entity has responsibility for starting and ending the game itself. Inside these functionalities, the condition checks and determinants of some hazards. The Game has a different concept from the Handler due to the fact that Game can do “operations” such as determining perihelion or not, creating the sun storm, and the like, rather than Handler handles each game object. The

possible concerned extensibility or requirement changes are following: changes the condition of winning/losing the game plays, changes from single player game to multiplayers game

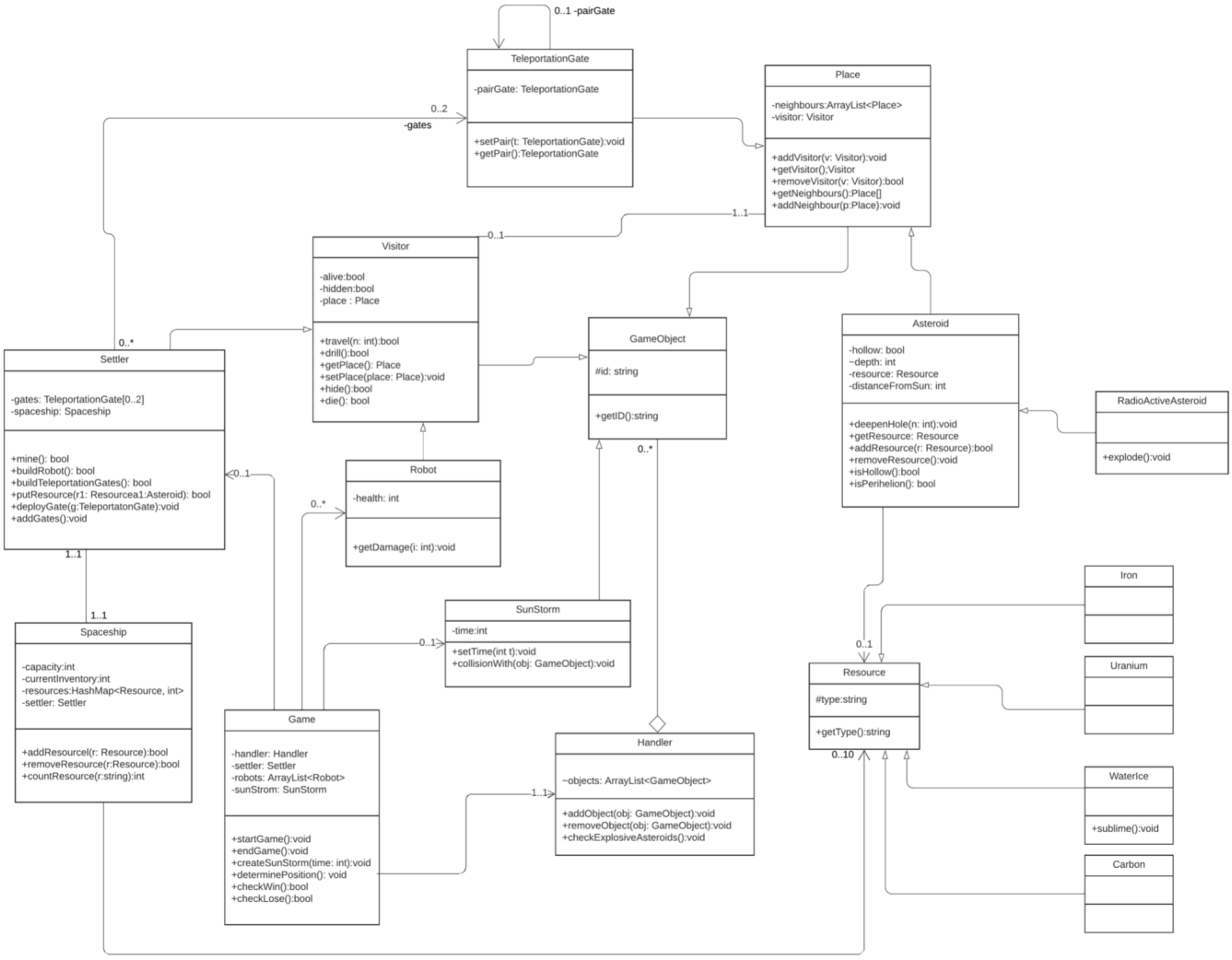
### 4.1.16. Handler

The Handler contains all necessary instances (Settler, Robot, Teleportation Gates, Asteroid) which relate to the played game. Moreover, it checks internal conditions of each game object (such as the game can know whether the asteroid is explosive or not) and add/removes GameObject in the game. This entity is essential because it must be possible to access all objects existing in the game in order to check the condition regarding events, and it makes the user able to notice the existence of the objects. The responsibility of this object is different from the Game, which is described above, because the former focuses on behaviors against the game objects, however, the latter deals with the game itself. The possible concerned extensibility or requirement changes are following: changes the computational/scientific method for collision (whether the collision occurred or not), changes number of objects that can exist in the game (e.g. up to 100 objects can appear in the game at same time)

### 4.1.17. Game Object

Game Object is an entity that has common functionalities among Visitor (Settler, Robot), Place (TeleportationGate, Asteroid) and Sun Storm. For example, unique id to distinguish, user view position, information needed for collision and extra computation are denoted here. This represents physical or graphical existence, namely, the concept of Game Object is differently categorized from logical existence such as Game, Handler or Resource because they are only values that the user cannot understand the meaning of them directly. The possible extensibility or requirement changes are following: extra information added to all of Visitor, Place and Sun Storm. Besides that game object can be considered as the higher level wrapper of the whole game which is used to manage the tasks and differentiate the individual entities inside the game.

## 4.2 Static structure diagrams




## 4.3 Class description

### 4.3.1. Visitor

#### Responsibility

The Visitor class is a parent class of Settler and Robot classes, since they share many attributes and features. It stores information about the asteroid the visitor is currently on (place), whether the visitor is hidden in a hollow asteroid, and if it is alive at all. It contains methods responsible for traveling, drilling, mining, and getting and setting the place of each visitor.

### 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), this condition will be later changed according to the events and things that take place.
- **bool hidden**: indicates if the visitor is hidden in the core of a hollow asteroid.It is an important attribute to understand the state of the settler as it can have limitations based on them.
- **Place place:** stores information about the place of the visitor(the asteroid that it is 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.
- **Place getPlace -** Gets the current position of a visitor.
- **void setPlace**- Sets the position of a visitor.
- **bool hide** - Checks if a visitor is hidden in a hollow asteroid. If not, the action of hiding a visitor is performed, and a true is returned.
- **bool die -** If the health of a robot reaches zero, it is killed, with no effect to the rest of the game. If a settler is not hidden in an explosion, it is killed, and the game ends.

## Superclasses

- **GameObject**

## 4.3.2 Settler

### Responsibility

The settler class inherits many attributes and methods from the visitor class. The Settler class has a few additional attributes and methods, since a Settler has some different capabilities and responsibilities compared to the other type of visitors-robots. This class stores information about the teleportation gates and spaceship belonging to a specific settler. It contains methods exclusive to settlers - such as mining, building robots and teleportation gates, storing resources and deploying gates.

### Attributes

- **TeleportationGate gates:** Stores information about any teleportation gates that the settler has built and deployed.
- **Spaceship spaceship:** Stores information about the personal spaceship of this settler that it uses for transportation.

### Methods

- **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 putResource** - The settler can fill a hollow asteroid with one unit of a resource.
- **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

- **Visitor->GameObject**

## 4.3.3 Robot

### Responsibility

This class is responsible for managing robots, which are a type of visitors. It stores information about the health of the robot, which is a metric indicating if a robot has been affected by dangers. It also contains a method that marks a robot as damaged, decreasing its health if it is affected by a danger.

### Attributes

- **int health:** the robot survives sun storms with damage only, so this is a metric of the damage made to the robot during a storm.

### Methods

- **void getDamage(int n):** In the case of a sun storm, a robot that was not hidden inside a hollow asteroid will be marked as damaged.

### Superclasses

- **Visitor->GameObject**

## 4.3.4 Spaceship

### Responsibility

The responsibility of this class is to store attributes regarding inventory, capacity, the resources stored in the spaceship, and the settler that the spaceship belongs to. The methods deal with inventory- adding resources, removing them, and counting them.

### 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.
- **int capacity:** Indicates the capacity of the spaceship when it comes to storing resources.
- **Settler settler:** Indicates which settler owns the spaceship.

### Methods

- **bool addResource (Resource r):** Checks if there is enough space for an additional unit of resource, and if so, adds it to the spaceship, and returns true.
- **bool removeResource(Resource r):** Removes a unit of resource from a spaceship and returns true.
- **int countResource(string r):** Counts the number of units of resource in a spaceship and returns an integer value.

## 4.3.5. Resource

### Responsibility

This class is a parent class to four different resources- uranium, iron, carbon and water ice. It contains the attribute type, that indicates the name/type, and a method to access that name and have it returned as a string.

### Attributes

- **string type:** Stores the name of the resource name/type.

### Methods

- **string getType:** Gets the name of the individual resource and returns it as a string.

## 4.3.6. Uranium

### Responsibility

The Uranium class is a child of the Resource class. It stores information about one type of the resources- Uranium.

## Superclasses

- **Resource**

## 4.3.7 Water ice

## Responsibility

Water ice is a child class of the Resource class. It stores information about one type of the resources- Water Ice. It also contains a method for water sublimation (disappearance).

## Methods

- **void sublime()**: When an asteroid with water ice in its core is at perihelion, this water ice sublimates(evaporates).

## Superclasses

- **Resource**

### 4.3.8 Iron

## Responsibility

Iron class is a child of the Resource class. It stores information about one type of the resources- Iron.

## Superclasses

- **Resource**

### 4.3.9 Carbon

### Responsibility

Carbon  class is a child of the Resource class. It stores information about one type of the resources- Carbon.

## Superclasses

- **Resource**

### 4.3.10 Place

## Responsibility

This class is a parent class of Asteroid and TeleportationGate. These two entities are places where a Visitor can be, and the Place class deals with them. It stores information about its neighbors, and the visitors of the places. It inherits its id from the GameObject class.

## Attributes

- **ArrayList<Place> neighbors:** All neighbors of a certain Place are stored in a list- neighbors are Asteroids or Teleportation Gates on either side of another Place.
- **Visitor visitor:** Stores information about which visitor is currently visiting a place.

## Methods

- **void addVisitor (Visitor v):** adds a visitor v as a current visitor to the place.
- **Visitor getVisitor:** gets information about which visitor is currently visiting a place, and returns it.

- **bool removeVisitor(Visitor v):** removes a current visitor when it leaves the place, and returns true.
- **Place[] getNeighbours:** gets neighbors of a certain place-the asteroids or teleportation gates on either side of it.
- **void addNeighbour(Place p):** adds a new neighbor to a place.

### Superclasses

- **GameObject**

## 4.3.11 Teleportation gate

### Responsibility

This class stores information about teleportation gates that were deployed by settlers. These come in pairs, so it is important for two gates of a pair to be linked with one another, which is why we need a pairGate attribute, and methods to set and get a gate's pair. This is a child class of the Place class, since it is one of the two available types of places, and through Place, it inherits its ID from Game Object.

### Attributes

- **TeleportationGate pairGate:** Stores information about what other gate is connected with this gate, since they come in pairs.

### Methods

- **void setPair(TeleportationGate t):** Assigns one gate to another, as they come in pairs.
- **TeleportationGate getPair():** Gets and returns the pair of a gate

### Superclasses

- **Place->GameObject**

## 4.3.12. Asteroid

### Responsibility

This class is a child class of Place, and through place it inherits an ID from Game Object. It is the parent class of RadioActiveAsteroid class, since radioactive asteroids are one distinct type of asteroid. This class stores information about whether asteroids are hollow, the depth of their

rock mantle, and which resource the core is made from. Also, it contains methods such as deepening the rock mantle, adding and removing resources, determining whether the asteroid is hollow, and whether or not it is at perihelion.

## Attributes

- **bool hollow:** stores information about whether an asteroid has a hollow core or not. A core is hollow when there is no resource.
- **int depth:** the depth of the rock mantle varies, and this attribute stories the depth of the mantle.
- **Resource resource:** stores information about the type of the resource which makes up the core of an asteroid.
- **int distanceFromSun:** Indicates the distance between a radioactive asteroid and the Sun. This is important to be able to determine when the asteroid is at aphelion or perihelion.

## Methods

- **void deepenHole(int n):** The rock mantle is deepened by one unit. The int parameter would be useful in case of requirement change, so the mantle could be deepened by a different amount.
- **Resource getResource():** Returns the name of the resource that makes up the asteroid core.
- **bool addResource(Resource r):** Adds resource r to the core of an asteroid and returns true if the action is performed successfully.
- **void removeResource():** Removes resource from the core of an asteroid.
- **bool isHollow():** Checks if an asteroid has a hollow core.
- **bool isPerihelion():** used to check if an asteroid is at perihelion position. Important for radioactive asteroids, and those with water ice in their core.

## Superclasses

- **Place->GameObject**

### 4.3.13 Radioactive Asteroid

## Responsibility

This is a child class of Asteroids. It is needed because one type of asteroid- those with uranium in their core are radioactive, and can explode when at perihelion. Since it’s a special behaviors of a particular asteroid and it as a whole retains properties of a normal asteroid we used inheritance

### Methods

- **void explode():** When an asteroid with a radioactive core is at perihelion, it will explode, killing settlers, and displacing robots.

### Superclasses

- **Asteroid->Place->GameObject**

## 4.3.14. Sun Storm

### Responsibility

This class stores information about the sun storm- when it occurs. It also contains the methods needed for the collisions with objects of the game. The sun storm inherits its ID from the Game object.

### Attributes

- **int time:** stores a time value for setting the time of sun storm occurrence.

### Methods

- **void setTime(int t):** Sets the time t when the sun storm will occur.
- **void collisionWith(GameObject obj):** Performs the collision of the sun storm with one of the objects in the game.

### Superclasses

- **GameObject**

## 4.3.15 Game

### 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

- **Handler handler:** 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.
- **ArrayList<Robot> robots:** Game system must know all robots since they are controlled by the system.
- **SunStorm sunStorm:** Sun storm object which is used when a sun storm occurs.

### 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(int time):** this function can be invoked randomly and creates an occurrence of a sunstorm in the game
- **void determinePosition():** Determines if an asteroid is at perihelion or aphelion, based on the attribute distance from Sun. This is important for water ice asteroids and radioactive asteroids.
- **bool checkWin():** Checks if the needed resources have been collected on an asteroid, and if so, the game is won.
- **bool checkLose():** Checks if the settler has been killed by an explosion, and if so, the game is lost.

## 4.3.16 Handler

### Responsibility

Handler is a container that contains all necessary instances related to the played game. A settler, asteroids, robots and teleportation gates are contained. Furthermore, this has the responsibility for checking whether the asteroid is explosive or not.

## 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.
- **void checkExplosiveAsteroids():** calls explode() if the radioactive asteroid is on the perihelion.

## 4.3.17 Game Object

### Responsibilities

GameObject is responsible for dealing with existing objects in the game. During the extension or requirement changes, we will add the functions which are commonly used among derived classes. Many classes inherit their unique ID from the Game Object class, since it is their superclass.

### Attributes

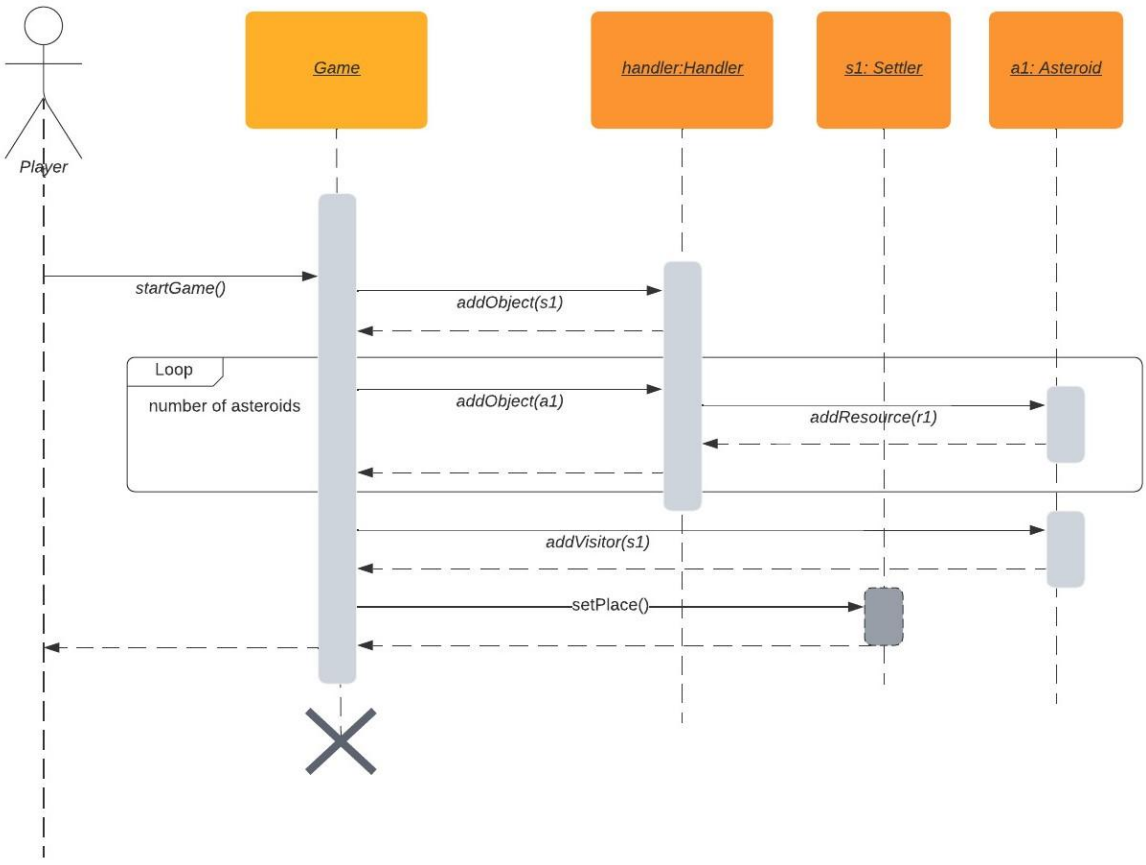
- **string id:** stores the unique ID of objects in the game

### Methods

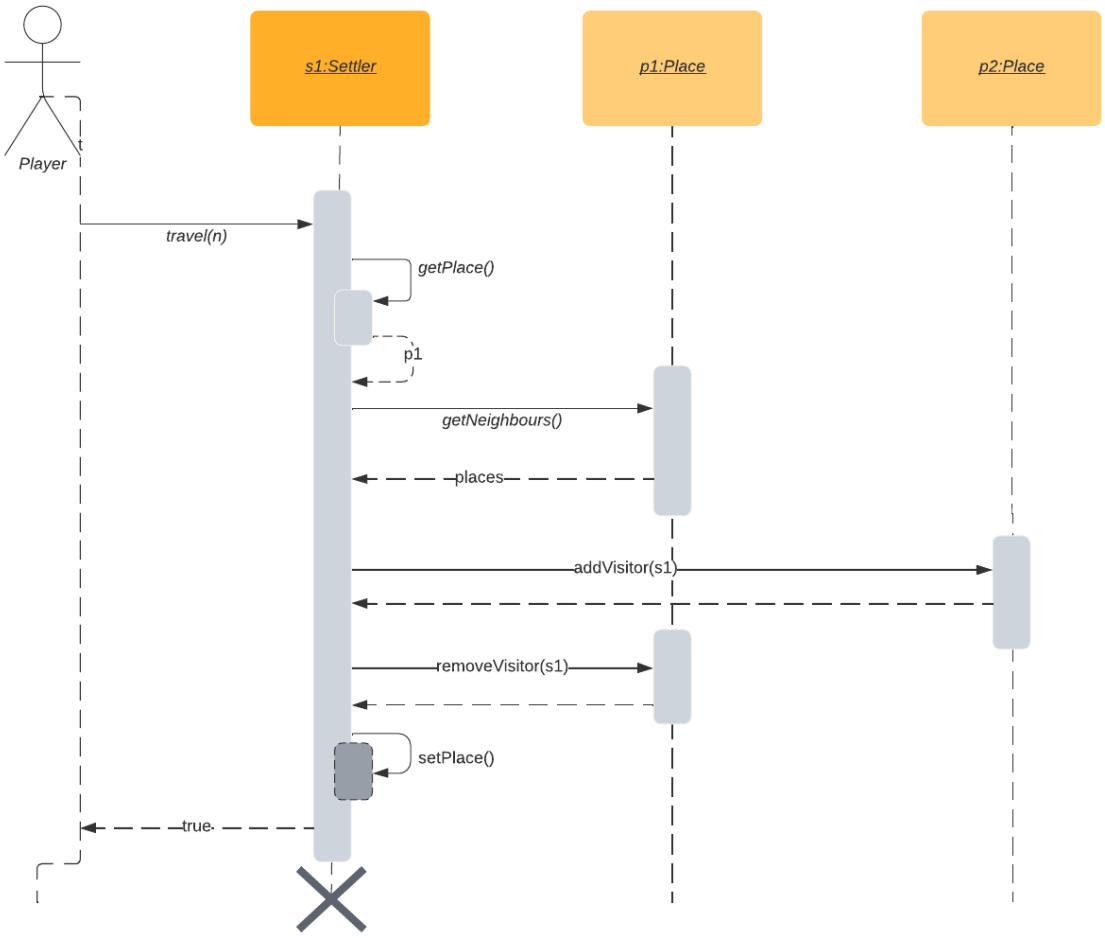
- **string getID():** gets the unique ID of an object

# 4.4 Sequence diagrams

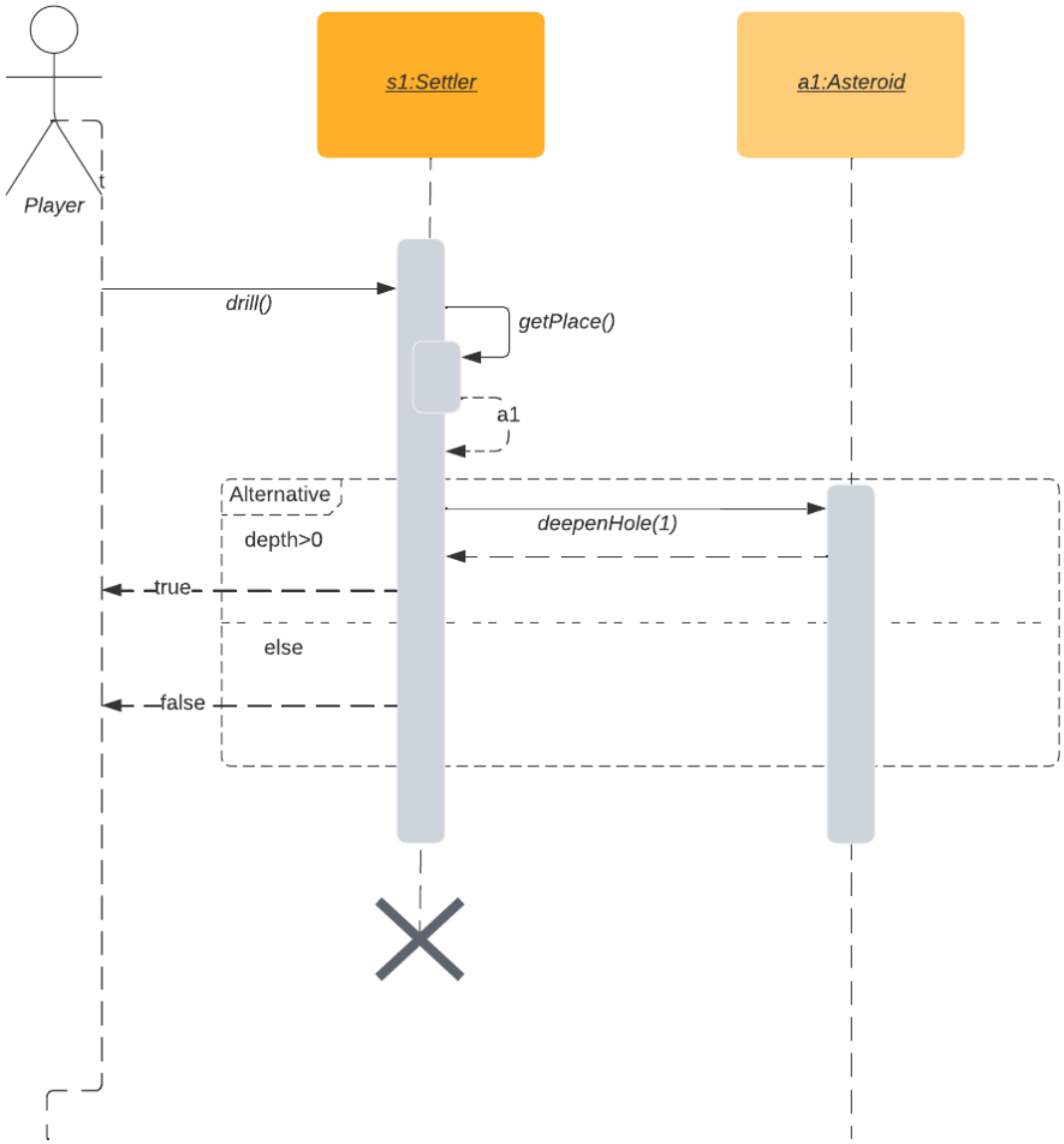
1. **Begin Game (Initialization & Settlers are positioned on the asteroids as start point of the game):**



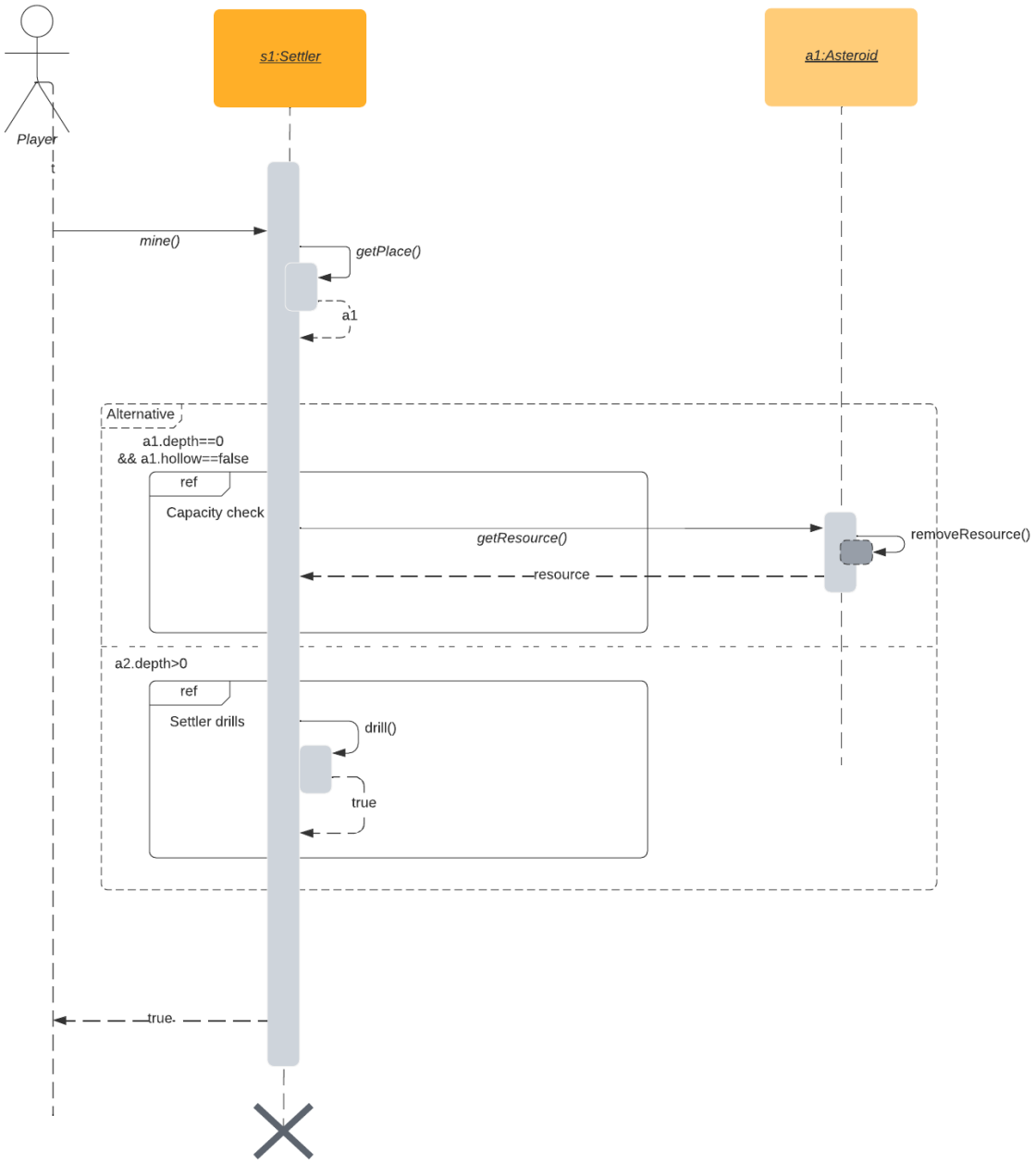
## 2. Settler moves



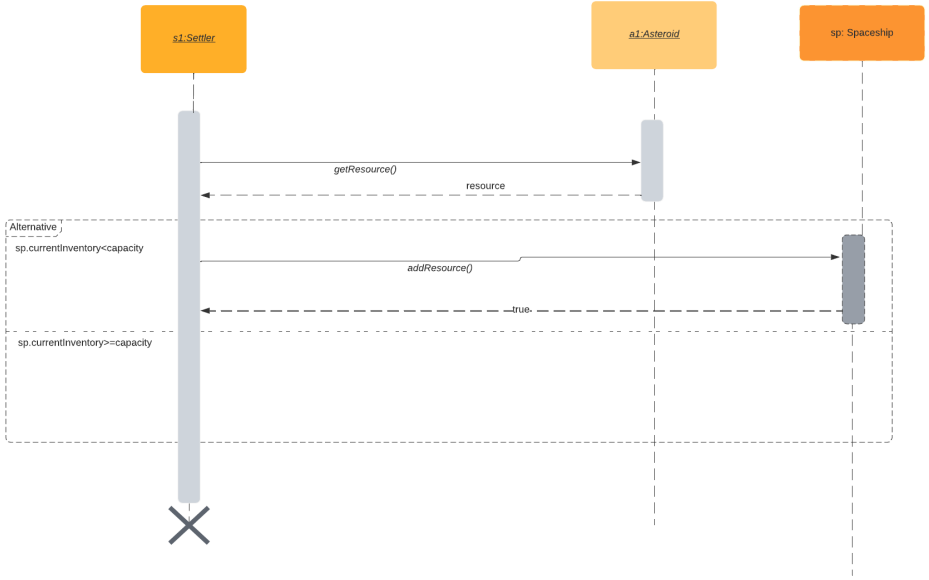
### 3. Settler Drills:



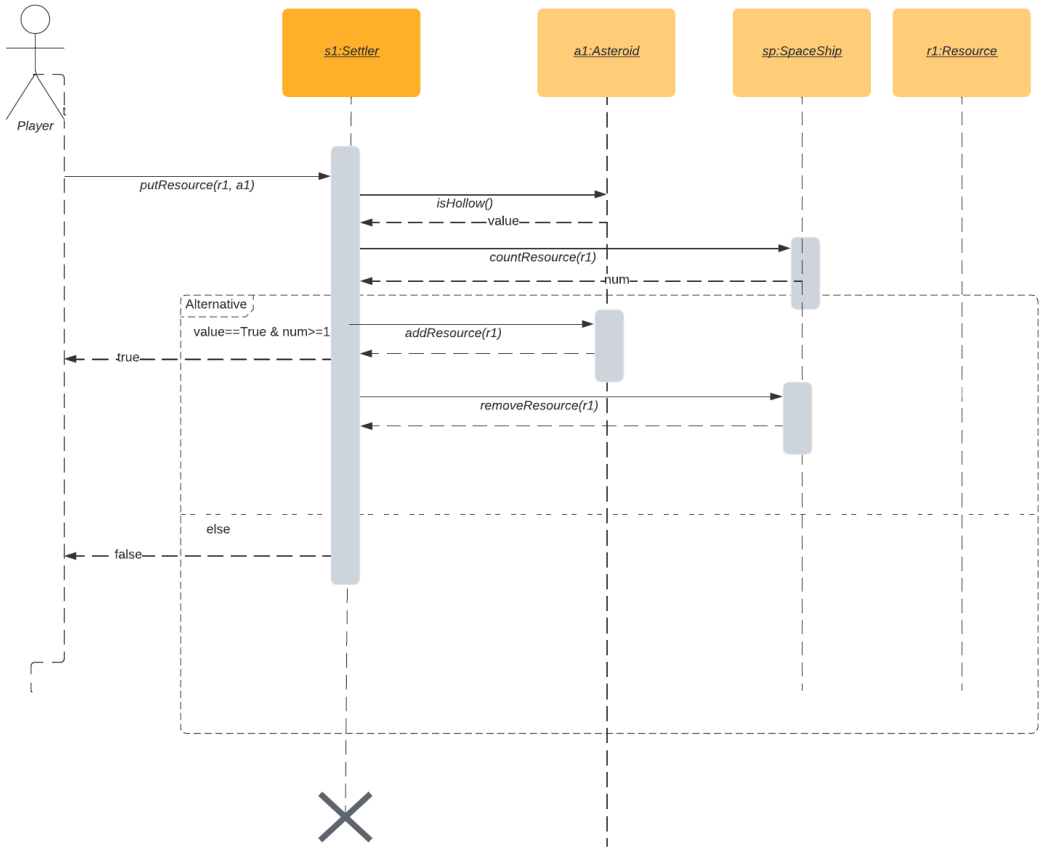
## 4. Settler Mines:



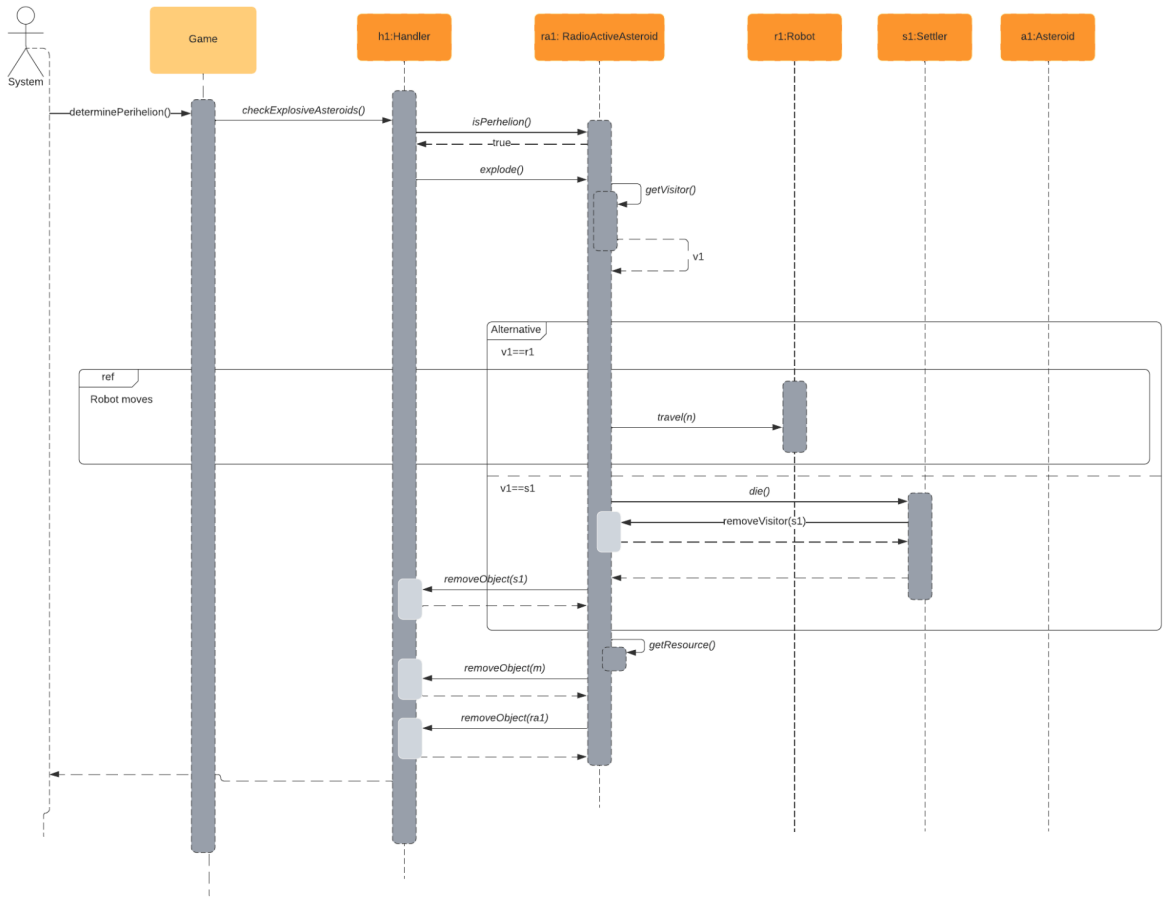
## 5. Capacity Check



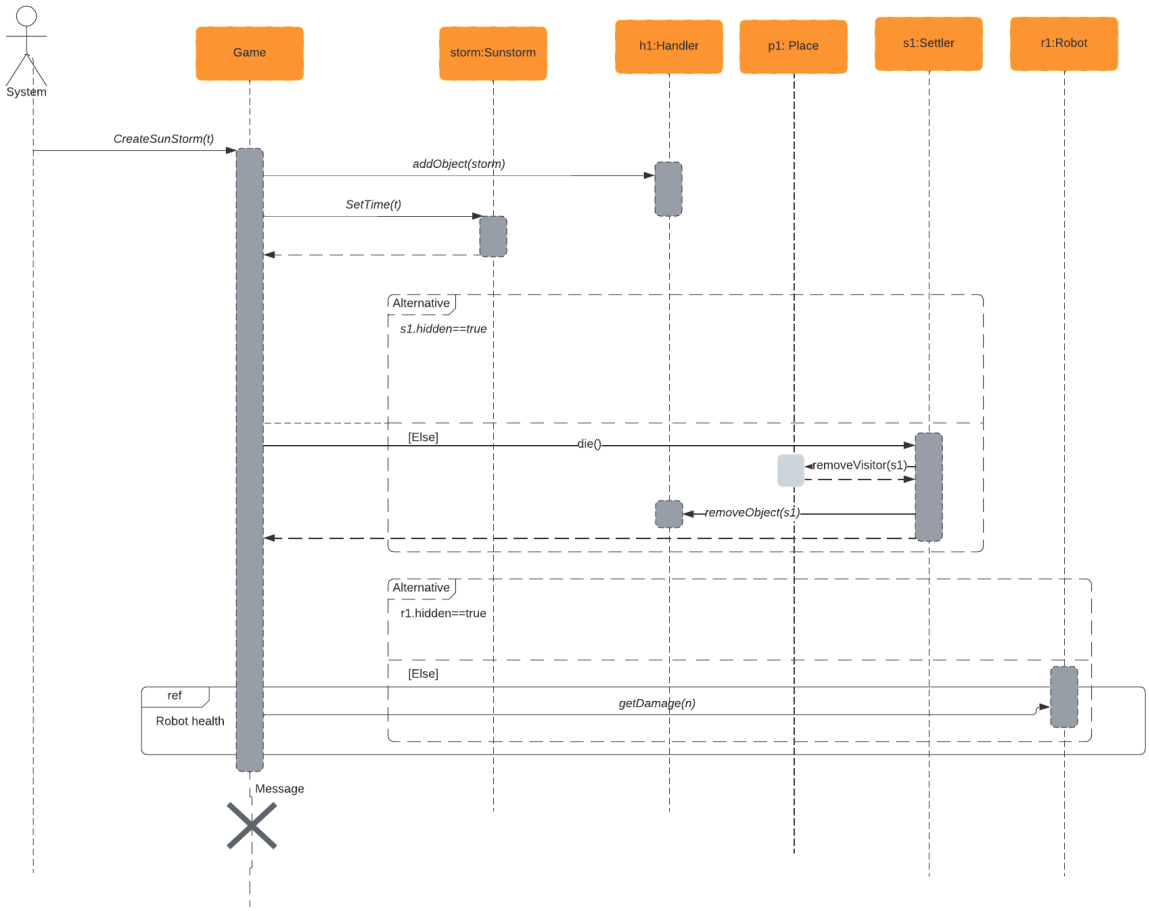

## 6. Settler puts resource down:



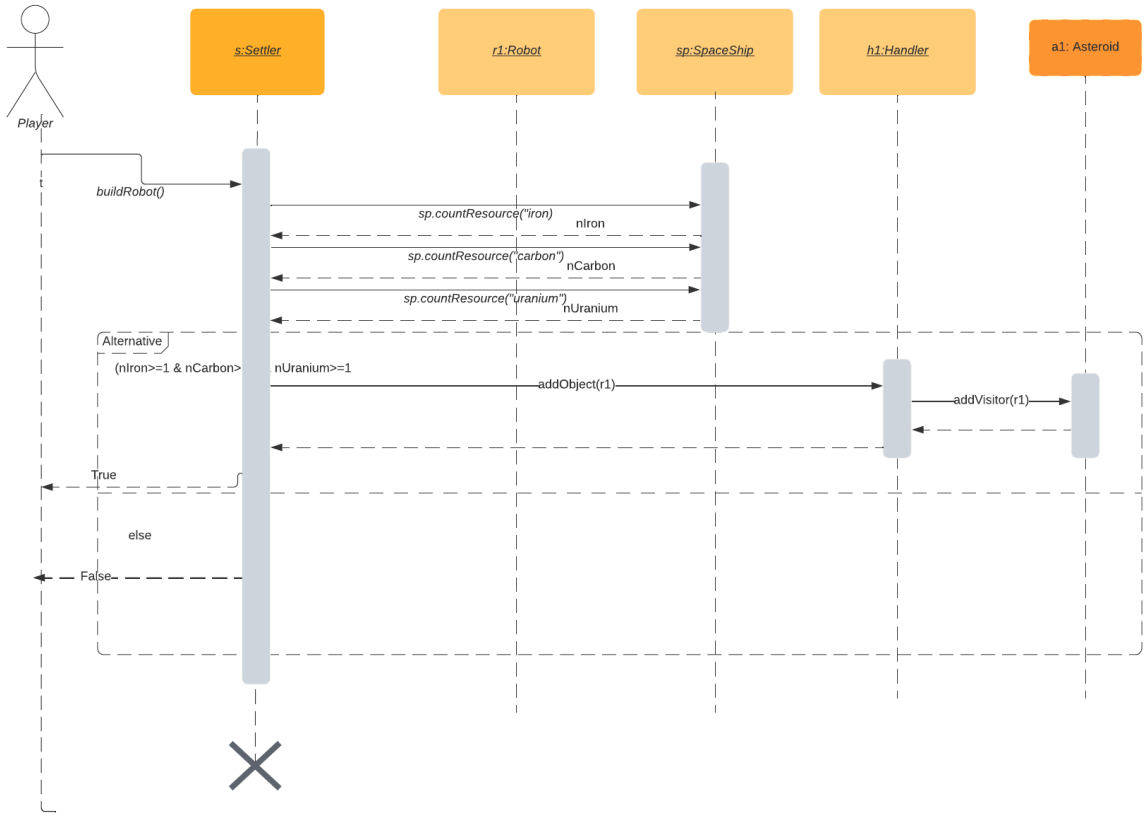
## 7. Asteroid Explodes (perihelion):



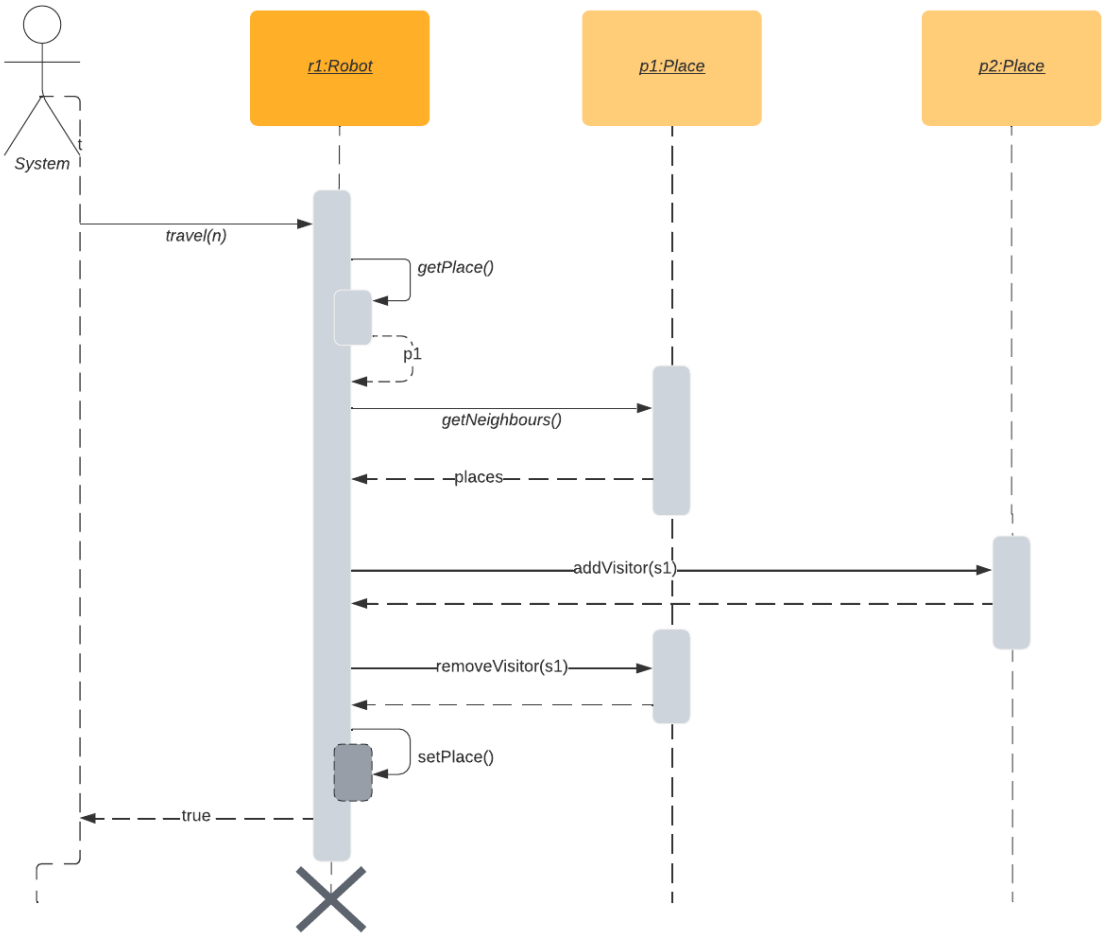
## 8. Sunstorm occurs:



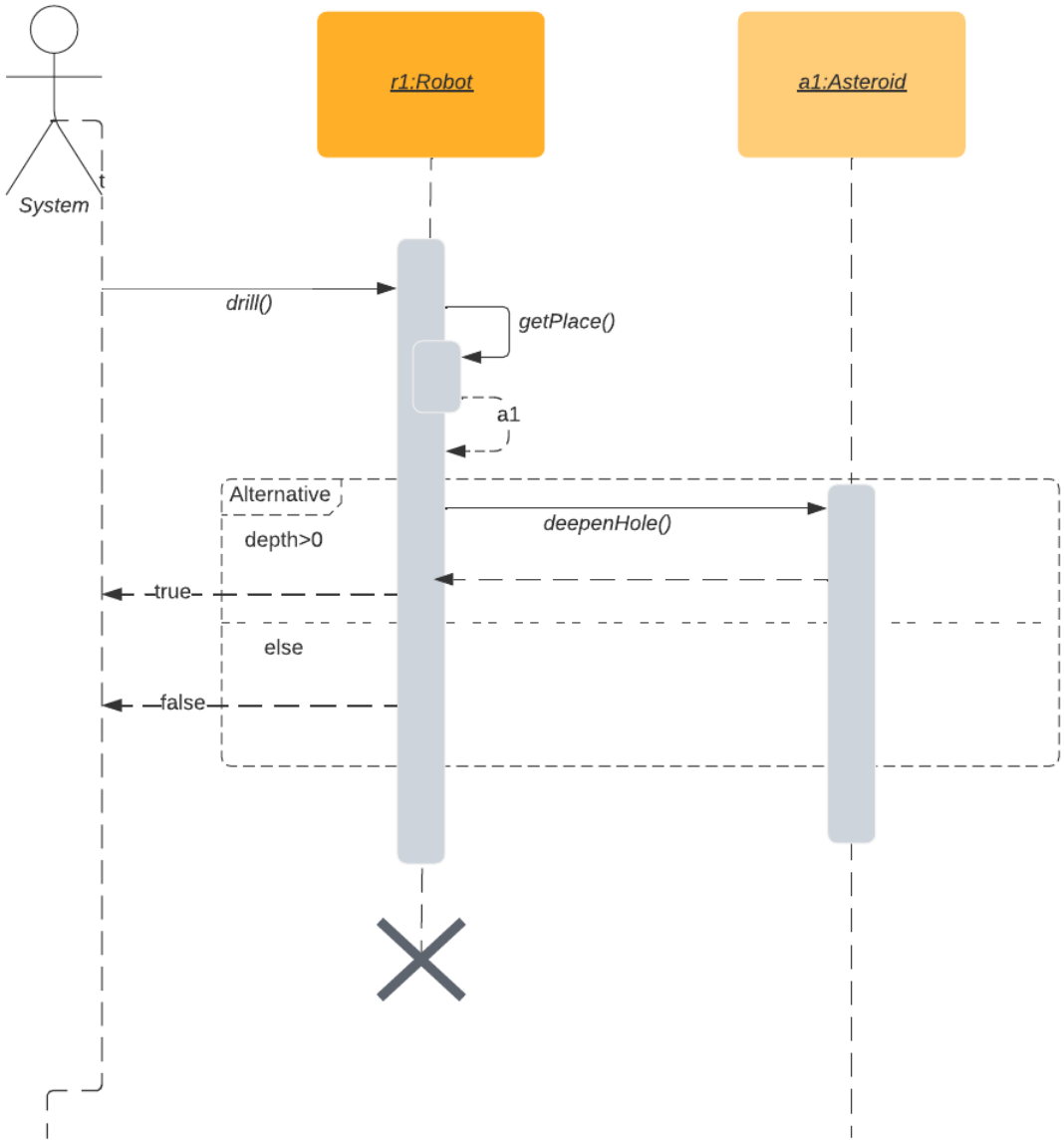
## 9. Robot is created:



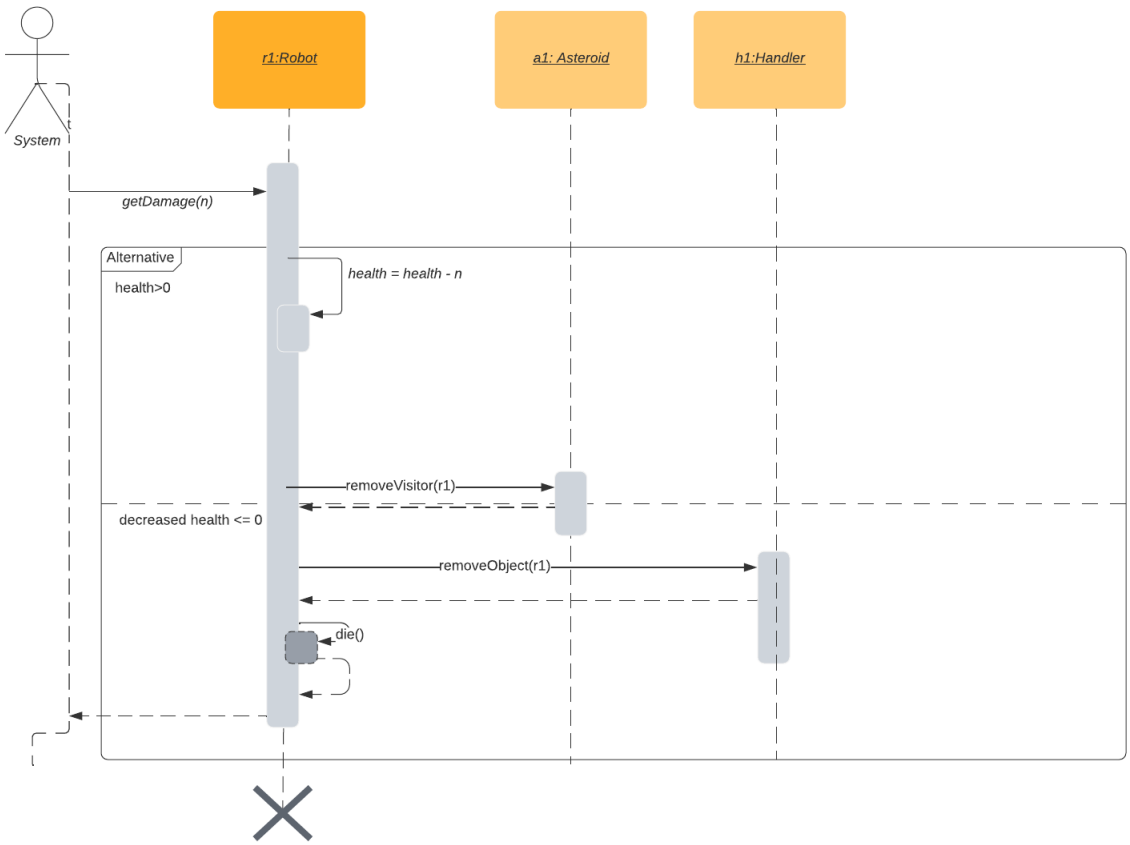
## 10. Robot moves:



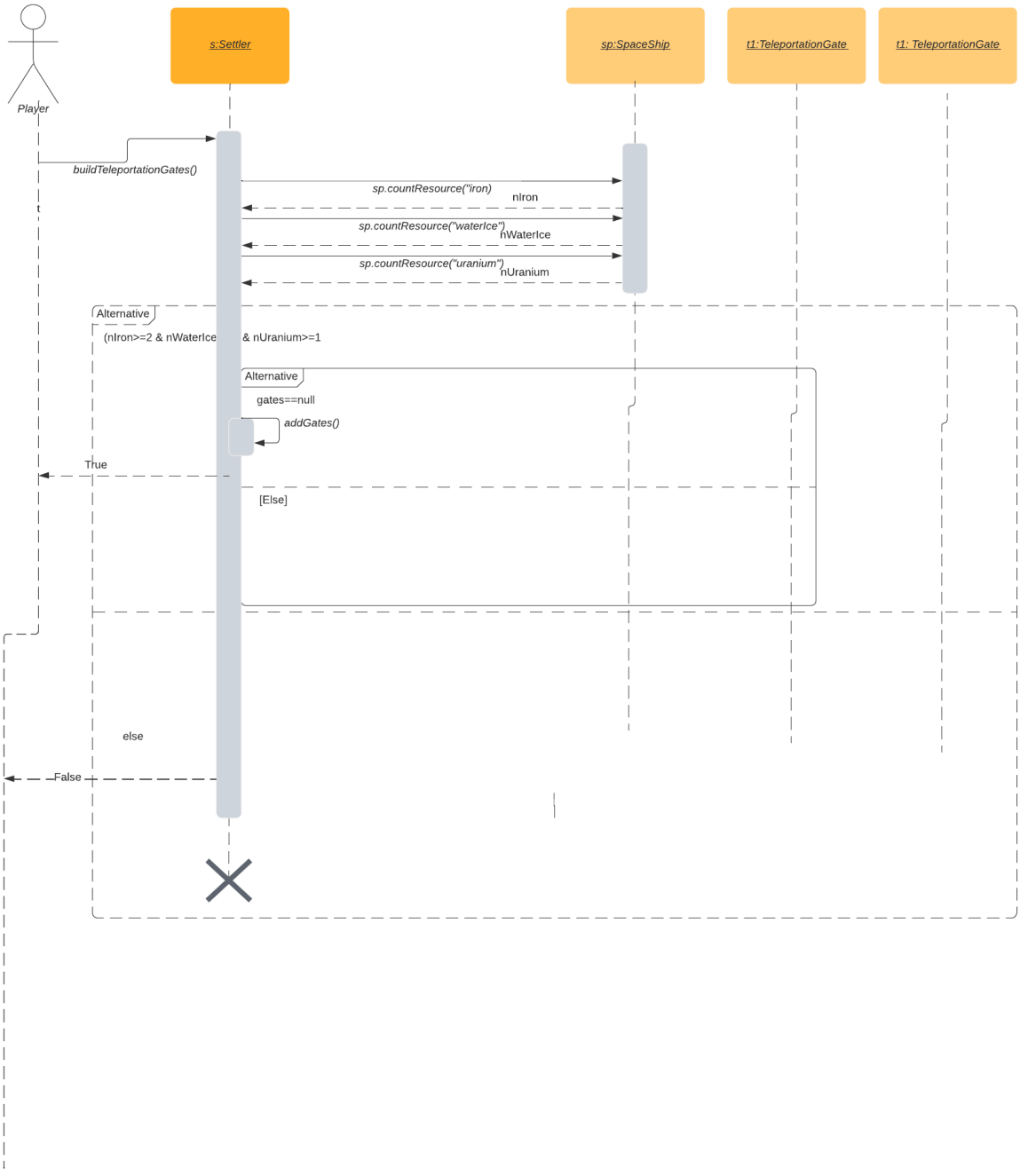
## 11. Robot drills:



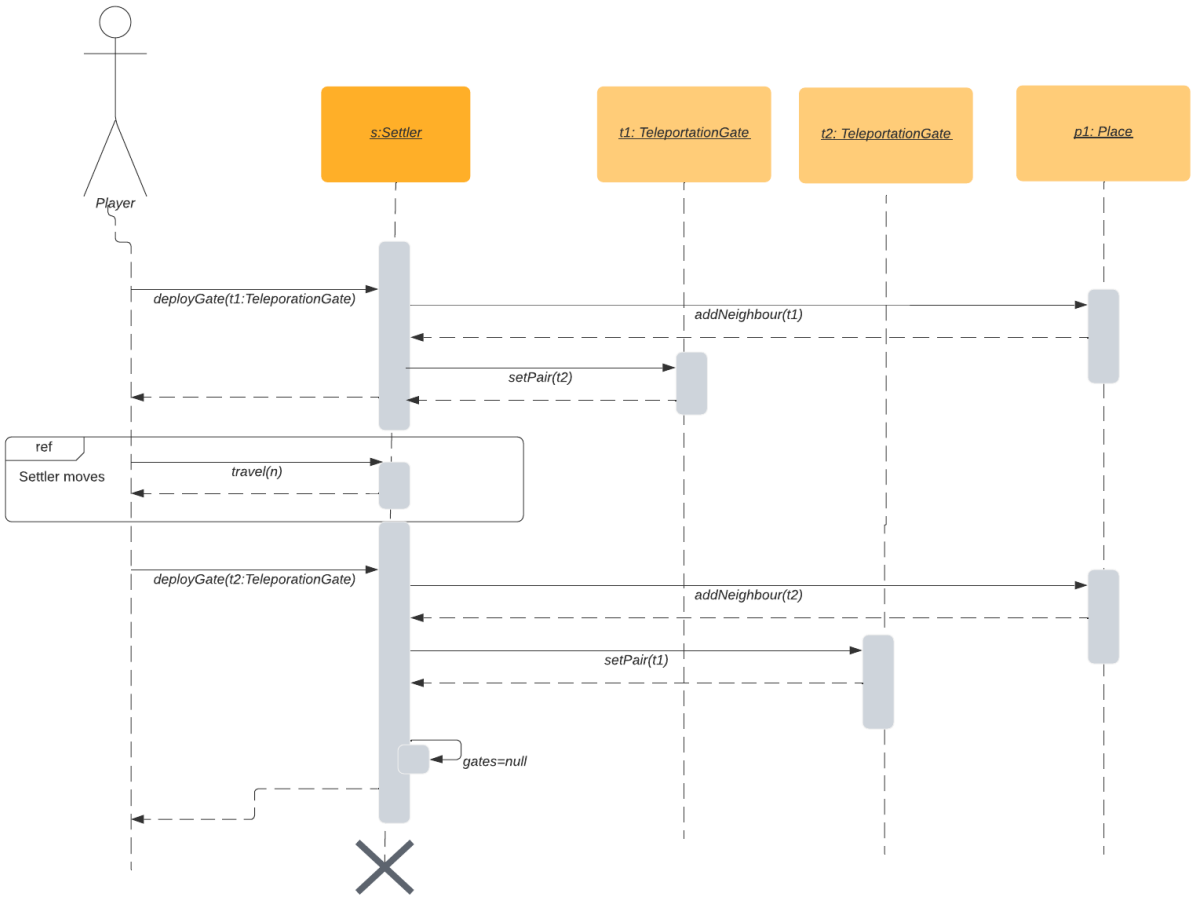
## 12. Robot health:



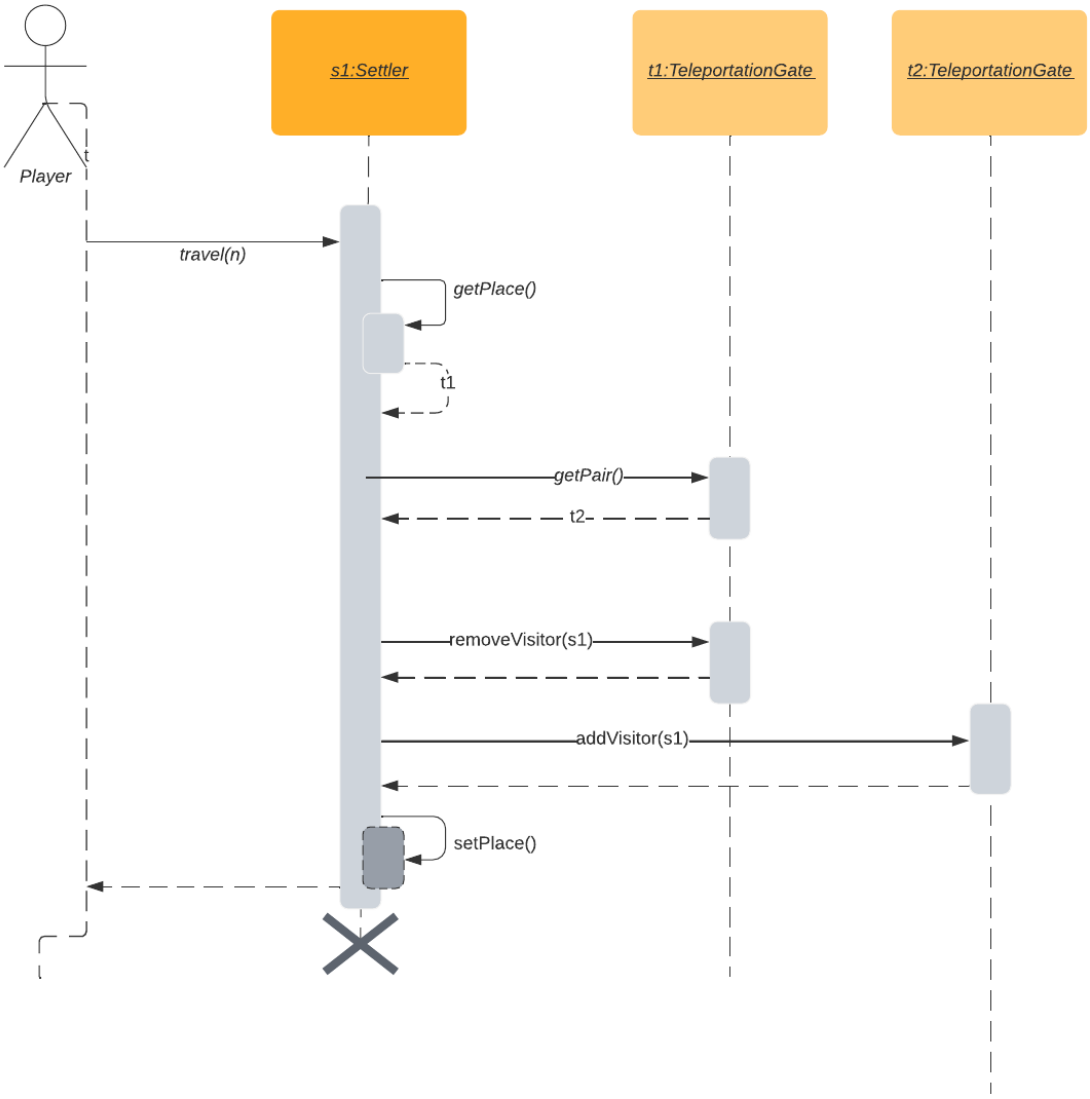
## 13. Pair of Teleportation-gates are created:



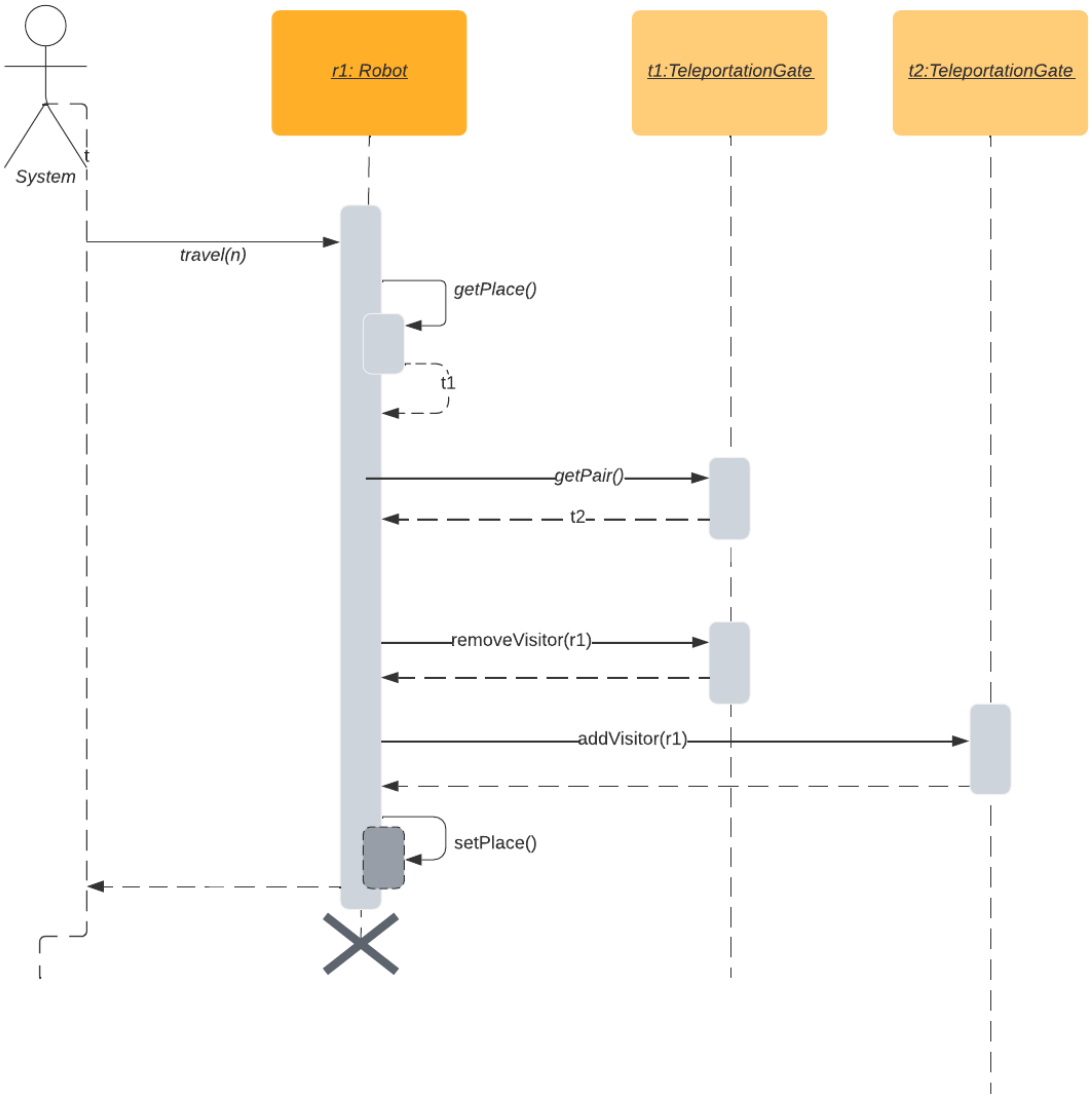
## 14. Teleportation gates are deployed:



### 15. Settler passes through teleportation-gate:

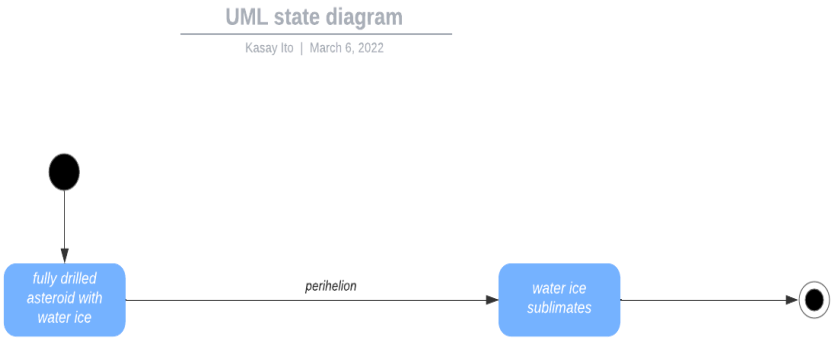


## 16. Robot passes through teleportation-gate:

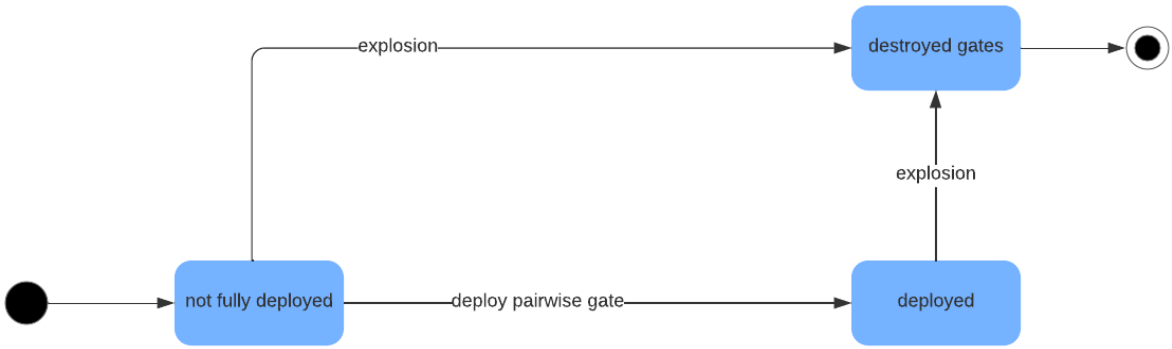


# 4.5 State-charts

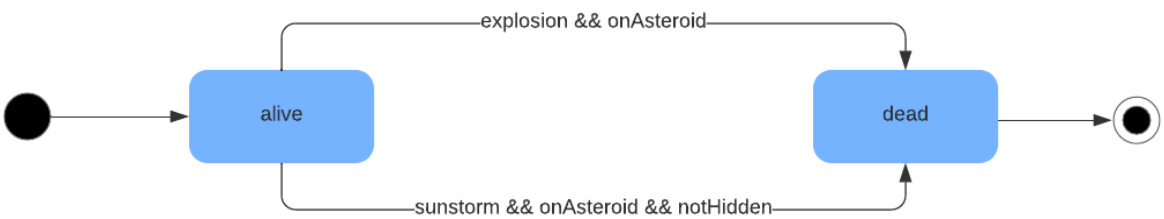
### 1. Ice sublimates in asteroid



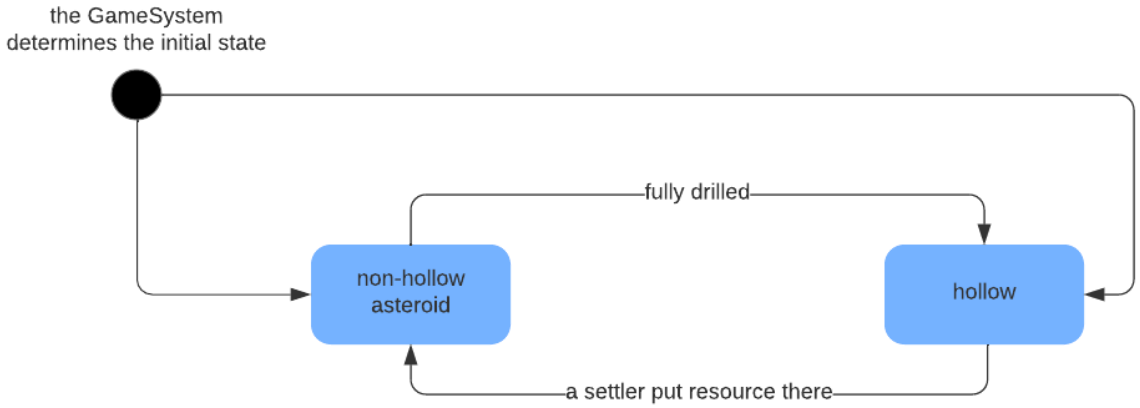

### 2. Teleportation gates deployment



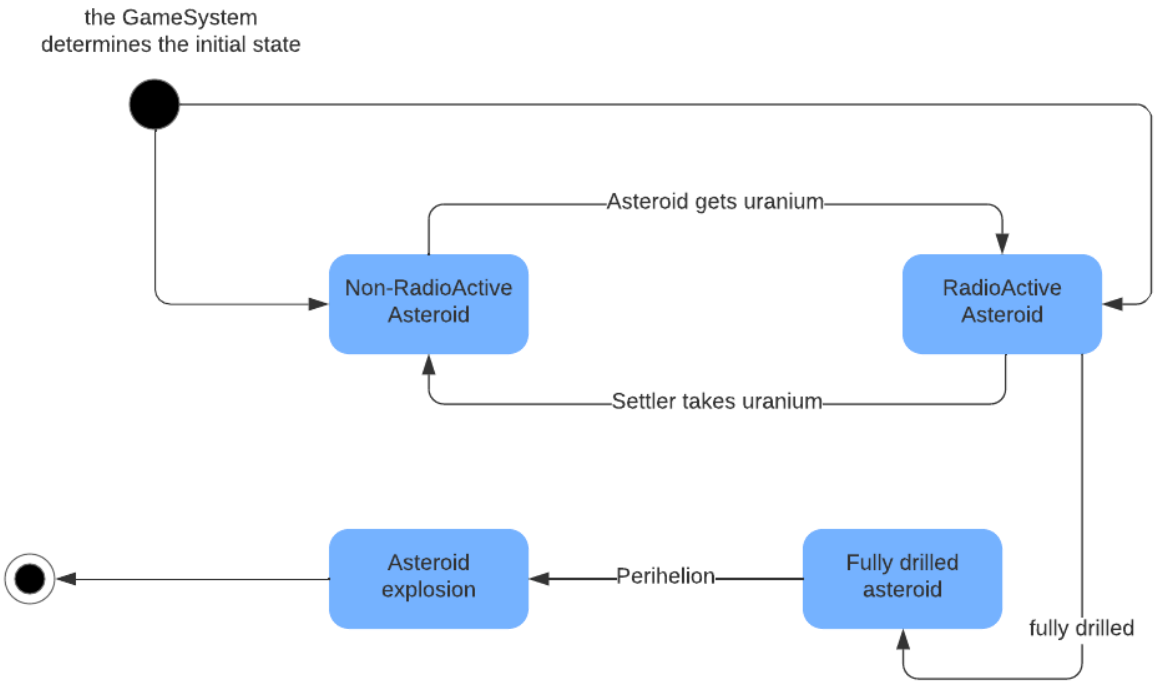

### 3. Settler lives



## 4. Hollow Asteroid

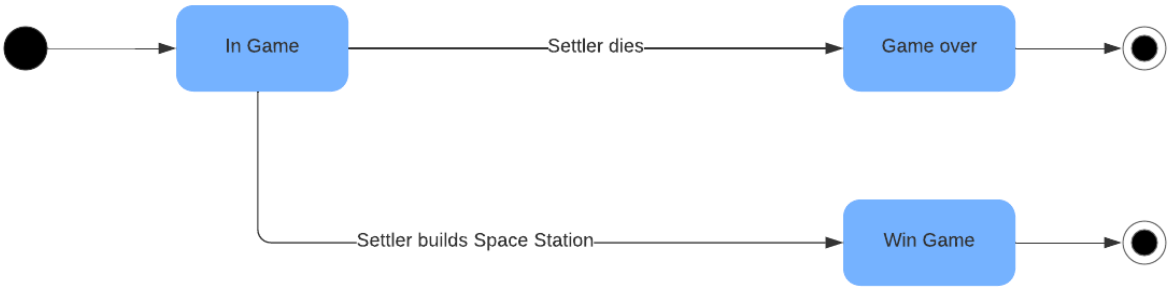



## 5. Asteroid explodes



### 6. Robot dies


### 7. Game ends




## 4.6 Protocol

| Start (date & time) | Duration (hours) | Performer(s) name | Activity description |
|---|---|---|---|
| 10/03/2022 | 4 hours | All team members | Class diagram refinement based on the feedback from the mentor, distributing tasks for each member - split into pairs that |

| | | | |
|---|---|---|---|
| Online meeting on teams | | | worked together, discussed what tool to use |
| 14/03/2022<br><br>Online meeting on teams | 4 hours | Janibyek,Chaitanya, Desoki | Made changes to the class diagram. And worked on the sequence diagram by drawing required diagrams from the professor. Checked consistency of methods among class and sequence diagrams. |
| 14/03/2022<br><br>Online meeting on teams | 2.5 hours | Kasay and Neda | Kasay and Neda had the responsibility of editing the documentation. Object catalog was finished, and class descriptions were started, alongside checking consistency of class/sequence diagrams. |
| 14/03/2022 | 2.5 hours | Kasay | Making small edits to class and sequence diagrams, properly labeling and ordering them, and adding them into the documentation. |
| 15/03/2022 | 3h | Tushig | We decided to use the holiday to tackle some future tasks in advance, so Tushig did research for potential game engines, and some UI design-background and asteroids. |

| | | | |
|---|---|---|---|
| 15/03/2022 | 2.5 hours | Neda and Tushig- | Finishing the class descriptions, and changing some details in the class diagram on the way. |
| 16/03/2022 | 1.5hours | Chaitanya | Started refining the contextual portion of the documentation. Stressing more upon the mentors' comments and highlighting the relevant changes we made and why they were necessary. |