3. Entwicklung des Analysemodells

2 – Hallooo Matematikeeer

Konzulent:
Márton Kovács

Mitglieder

Henrietta Domokos	J0DCPT	domoheni@gmail.com
Ábel Borbély	DRP0DT	babel1122@gmail.com
Renátó Hrotkó	OIT6HD	renatohrotko@gmail.com
Vince Pongrácz	MKMDJO	pongrvin@gmail.com

17.02.2021

3. Entwicklung des Analysemodells

3.1 Objekten Kataloge

3.1.1 Asteroid
Daraus können die Spieler Ressourcen fördern nachdem ihre Mantel durchgebohrt wurde. In Sonnennah kann es explodieren falls radioaktive Ressource enthält und der Spieler es versuchen hat durchzubohren.

3.1.2 Ressources
Es enthält die Typen der möglichen Ressourcen die während des Spiels gefördert werden können. Es ist auch gewusst ob es radioaktiv ist oder nicht.

3.1.3 Core
In einem Asteroide ist ein Core/Kern der enthält immer ein Ressource. In dem Kern ist immer nur ein Ressource außer der Basisasteroide der viel mehr speichern kann weil dort wird der Basis aufgebaut.

3.1.4 Layer
Jede Asteroide hat ein Mantel, dessen Dicke veränderlich ist. Beim bohren wird es gebohrt um das Rohstoff in dem Kern erreichen zu können. Beim Bohren wird es immer 1 Einheit kleiner.

3.1.5 Position
Es hat immer eine x und y Koordinate mit denen die Positionen der Entitäten, Asteroiden und der Sonne bestimmt ist. Asteroiden haben auch ein Radius drin mit dem bestimmt ist, wie nah andere Asteroiden sein können.

3.1.6 Entity
Die Spieler haben Entitäten zB: Siedler und AIRoboter. Ein Entität kann sich bewegen zwischen Asteroiden, bohren die Mantel der Asteroiden, sterben bei einer Explosion (außer Roboter) und Sonnenflair, die Nachbarn behandeln.

3.1.7 AIRoboter
Spieler können aus bestimmten Ressourcen Roboter herstellen die beim Bohren helfen können. Sie überleben den Explosion und landen auf einem benachbarten Asteroide. Sie sind von dem GameController kontrolliert also der Spieler kann mit ihnen nichts tun. Gleichzeitig kann ein Spieler mehr besitzen.

3.1.8 Settler
Spieler können sogar mehrere Siedler haben und mit ihnen können sie spielen. Sie können max 10 Ressourcen haben. Sie können noch fördern aus Asteroiden. Portals herstellen und bauen, Ressourcen zurücksetzen in Asteroiden falls sie leer sind.

3.1.9 Player
Die Spieler haben immer ein oder mehr Siedler und können mehrere Roboter besitzen. Sie haben auch ein Name der vor dem Spiel eingestellt wird.

3.1.10 Sun
In einem Asteroidenzone ist immer eine Sonne. Sie kann ein Solarflair machen, das Die Siedler und Roboter nur in einem leeren Asteroide überleben können.

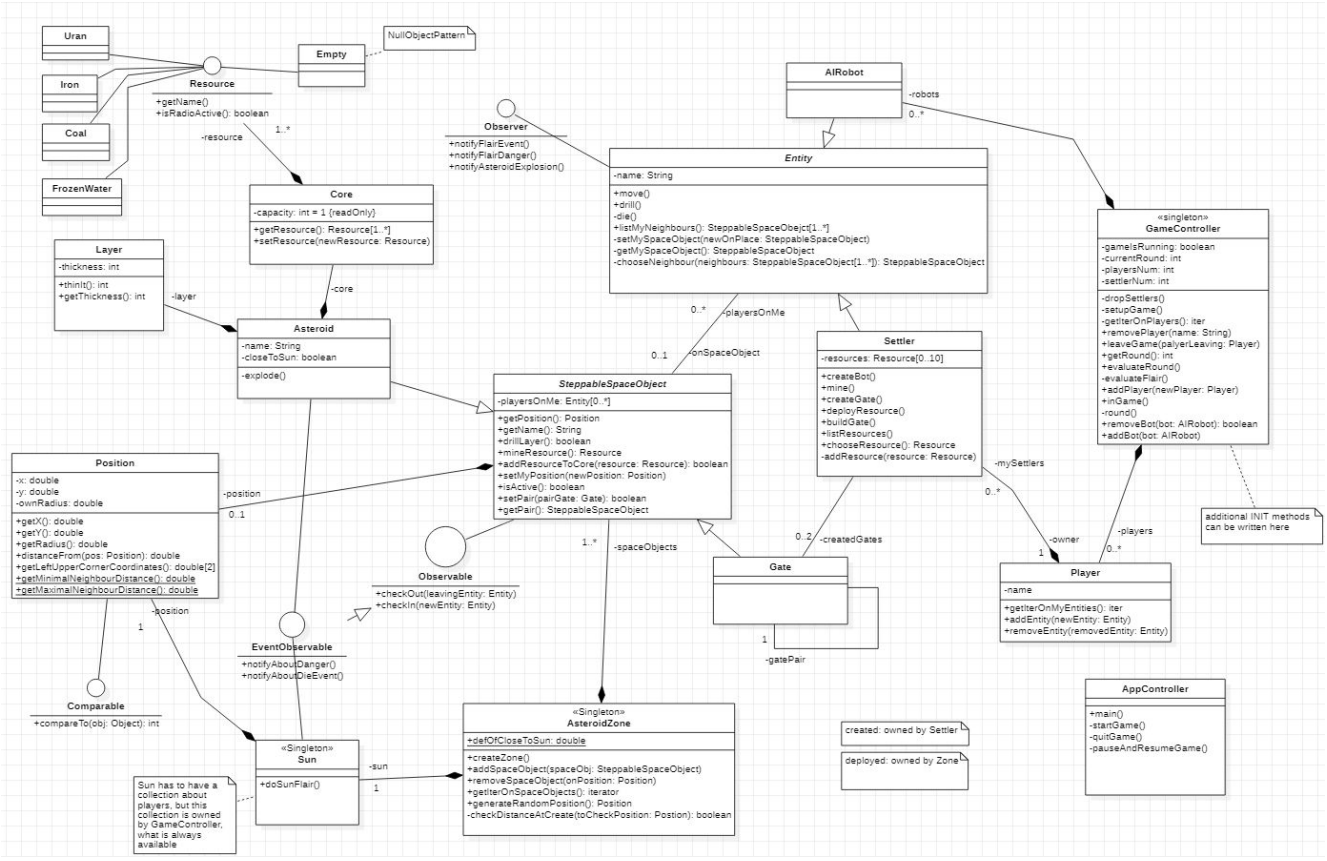
3.1.11 Gate
Spieler können über Portale verfügen um nach einem anderen Ort zu teleportieren. Ein Spieler kann gleichzeitig immer nur ein Portalpaar dabeihaben.

3.1.12 AsteroidZone
Es kann die ganze Asteroidenzone herstellen. Also die Spieler, Sonne, Asteroiden positionieren mit zufälliger Verteilung bei den Asteroiden.

3.1.13 GameController
Es kontrolliert den ganzen Spiel. Die Runde die nacheinander kommen sind hier behandelt und am Ende ausgewertet. Die AIRoboter sind auch von ihm kontrolliert. Es kann den Spiel einstellen.

3.1.14 AppController
Ein AppController kann den ganzen Spiel laufen lassen, während dem Spiel pausieren und dann fortsetzen und falls der Spieler möchte dann den ganzen Spiel sofort beenden.

3.2 Statische Struktur Diagramme



3.3 Beschreibung der Klassen

3.3.1 AIRobot

- **Verantwortung**

  Der AIRobot bohrt, bewegt sich und hilft den Spielern, ihr Ziel zu erreichen.

- **Basisklasse**

  Entity → AIRobot

3.3.2 AppController

- **Verantwortung**

  Diese Abteilung ist für die Ausführung des Spiels verantwortlich. Startet, stoppt oder pausiert das Spiel.

- **Methoden**
  - **main()**: Ein Programm muss eine globale Funktion namens main enthalten, die den festgelegten Start des Programms darstellt.

3.3.3 Asteroid

- **Verantwortung**

  Der Asteroid speichert Rohmaterial in seinem Kern. Er kann Astronauten schützen, wenn sie sich in seinem Kern verstecken. Das Asteroid kann explodieren. Das Rohmaterial des Kerns kann abgebaut werden..

- **Ősosztályok**

  SteppableSpaceObject → Asteroid

- **Schnittstellen / Interface**
  - EventObservable
  - Observable

- **Attribute**
  - **String name**: Der Name des Asteroiden.
  - **boolean closeToSun**: Ist der Asteroid in der Nähe des Sonne.

- **Methoden**
  - noch überschreibende Methoden von der abstrakten Basisklasse (SteppableSpaceObject)
  - **explode():** innere Method um Explosion zu behandeln

3.3.4 AsteroidZone

- **Verantwortung**

  Verwaltet das Spielfeld.

- **Attribute**
  - **int defOfCloseToSun**: Wenn Sie diese Zahl mit einem SteppableSpaceObject vergleichen, können Sie feststellen, ob es sich in der Nähe der Sonne befindet

- **Methoden**
  - **createZone()**: Es schafft das Spielfeld vor dem Spielstart.
  - **addSpaceObject(spaceObj: SteppableSpaceObject)**: Fügt der Asteroidenzone einen SteppableSpaceObject hinzu.
  - **removeSpaceObject(onPosition: Position)**: Entfernt ein Objekt in einer Asteroidenzone von einer bestimmten Position.
  - **iterator getIterOnSpaceObjects()**: Wenn ich die Nachbarn auflisten möchte, gibt diese Funktion eine Liste aller SpaceObjects und ich kann basierend darauf weiter filtern.
  - **Position generateRandomPosition()**: Erstellt Positionen durch Generieren von Zufallszahlen. Erforderlich, um die Strecke zu bauen.

3.3.5 Core

- **Verantwortung**

  Es ist verantwortlich für die Speicherung der Rohstoffe innerhalb des Asteroiden

- **Attribute**
  - **int capacity**: Gibt an, wie viele Rohstoffe sich im Kern befinden können.

- **Methoden**
  - **Resource[1…*] getResource**: Es kehrt mit dem Rohmaterial des Kerns zurück.
  - **setResource(newResource: Resource)**: Setzt den Kernrohstoff auf eine bestimmte Ressource.

3.3.6 Entity

- **Verantwortung**

  Abstrakte Basisklasse für Roboter und Siedler.

- **Schnittstellen / Interface**
  - Observer → Entity

- **Attribute**
  - **String name**: Name des Entity.

- **Methoden**
  - **move()**: Durch Aufrufen dieser Funktion werden die Entitäten verschoben.
  - **drill()**: Durch Aufrufen dieser Funktion werden die Entitäten bohren.
  - **SteppableSpaceObject[] listMyNeighbours()**: Gibt die Nachbarn des Asteroiden zurück, auf dem sich die Entität befindet.

3.3.7 EventObservable

- **Verantwortung**

  Wenn ein Asteroid explodiert oder eine Sonneneruption gibt, benachrichtigt SteppableSpaceObject.

- **Methoden**
  - **notifyAboutDanger():** Informiert alle aufgeschriebene Entität über die Gefahr von Solarflair.
  - **notifyAboutDieEvent():** Informiert alle aufgeschriebene Entität über den Tod, und das Grund darauf.

3.3.8 GameController

- **Verantwortung**

  Verwaltet den Verlauf des Spiels. Vom Start zum Ende.

- **Attribute**
  - **boolean gameIsRunning**: Sagt dir, ob das Spiel läuft.
  - **int currentRound**: Die Nummer der aktuellen Runde des Spiels.
  - **int playersNum**: Anzahl der Spieler. Etwas, das während des Setups konfiguriert werden kann.
  - **int settlerNum**: Anzahl der Siedler. Etwas, das während des Setups konfiguriert werden kann.

- **Methoden**
  - **removePlayer(name: String)**: Spieler aus dem Spiel entfernen.
  - **leaveGame(playerLeaving: Player)**: Wenn ein Spieler aufhören möchte, gibt er auf, tötet im Wesentlichen alle seine Siedler und löscht sie auch aus den Spielern.
  - **int getRound():** Gibt das Anzahl der aktuelle Runde im Spiel zurück.
  - **evaluateRound():** Wertet den gegebenen Kreis aus. Überprüfen Sie, ob alle Siedler des Spielers gestorben sind, und ob die Spieler gewonnen oder verloren haben.
  - **addPlayer(newPlayer: Player):** Fügt dem Spiel einen neuen Spieler hinzu.
  - **inGame():** Eine fast unendliche Schleife, in dieser Funktion läuft das Spiel und das Spiel kreist als Endlosschleife. Wir rufen diese Funktion auf, indem wir das Spiel starten und am Ende des Spiels beenden.
  - **boolean removeBot(bot: AIRobot):** Entfernt den Roboter. Der Boolesche Wert gibt an, ob Sie ihn entfernt haben.
  - **addBot(bot: AIRobot):** Fügt dem Spiel einen neuen AIRoboter hinzu.

3.3.9 Gate

- **Verantwortung**

  Kann von Siedlern erstellt und gespeichert werden. Ermöglicht das Reisen zwischen einem Paar von ihnen.

- **Basisklasse**

  SteppableSpaceObject → Gate

- **Attribute**
  - **Gate gatePair:** Der Paar eines Portals.

- **Methoden**

  nur überschreibende Methoden von der abstrakten Basisklasse.

3.3.10 Layer

- **Verantwortung**

  Es stellt die Kortikalis/Kruste/Layer des Asteroiden dar, weiß, wie dick er ist und kann ihn reduzieren.

- **Attribute**
  - **int thickness**: Die Dicke der Kruste des Asteroiden.

- **Methoden**
  - **int thinIt()**: Es verdünnt sich. Der DrillLayer ruft Sie an.
  - **int getThickness()**: Der Asteroid kehrt mit der Dicke seiner Kortikalis zurück.

3.3.11 Player

- **Verantwortung**

  Hilft bei der Verwaltung der Entitäten eines Spielers.

- **Attribute**
  - **String name:** Der Name des Spielers.
  - **Settler[0…*] mySettlers:** Die Kollektion der Siedler, die zu einem Spieler gehören.

- **Methoden**
  - **Iterator getIterOnMyEntities()**: Erzeugt einen Iterator für die Kollektion der Siedler eines Spielers
  - **addEntity(newEntity: Entity):** Weist einem Spieler eine neu Entität zu.
  - **removeEntity(removedEntity: Entity):** Hebt die Zuordnung einer Entität zu einem Spieler auf

3.3.12 Position

- **Verantwortung**

  Ordnet Objekten im Feld eine bestimmte Position zu.

- **Schnittstellen / Interface**

  Comparable

- **Attribute**
  - **double x:** Koordinate auf der x-Achse.
  - **double y:** Koordinate auf der y-Achse
  - **double ownRadius:** Hilft überlappende Objekte zu vermeiden.

- **Methoden**
  - **double getX():** Gibt die x-Koordinate zurück.
  - **double getY():** Gibt die y-Koordinate zurück.
  - **double getRadius():** Gibt den Radius des zugehörenden Objekts zurück.

- **double distanceFrom(pos: Position):** Kalkuliert die Distanz zweier Positionen.
- **double[2] getLeftUpperCornerCoordinate():** Hilft mit dem korrekten Platzieren der Objekte.
- **double getMinimalNeighbourDistance():** Untere Schranke für die Zufallsdistanz, in der die Objekte benachbart sind.
- **double getMaximalNeighbourDistance():** Obere Schranke…

3.3.13 Settler

- **Verantwortung**

  Siedler sind die Entitäten, die die Spieler bewegen und über sie mit dem Spielfeld interagieren können.

- **Basisklasse**

  Entity → Settler

- **Attribute**
  - **Resources[0...10] resources:** Speichert die abgebauten Ressourcen eines Siedlers.
  - **Player owner:** Der Spieler, dem die Siedler gehören.
  - **Gate[0...2] createdGates:** Die Kollektion der erstellten Portale.

- **Methoden**
  - **createBot():** Erschafft einen AI Roboter.
  - **mine():** Entfernt die Ressource aus dem Asteroidenkern, und fügt es seiner eigenen Kollektion von Ressourcen zu.
  - **createGate():** Erstellt ein Paar Portale, und fügt sie seiner eigenen Kollektion von Portalen zu.
  - **deployResource():** Setzt eine Ressource wieder in einen Asteroiden ein.
  - **buildGate():** Platziert eines der getragenen Portale.
  - **listResources():** Listet die Ressourcen auf, die der Siedler besitzt.
  - **Resource chooseResource():** Der Siedler wählt eine Ressource aus seinem Inventar aus.
  - **addResource(resource: Resource):** Fügt Ressource zu der Kollektion .

3.3.14 SteppableSpaceObject

- **Verantwortung**

  Diese sind Objekte, auf die die Spieler mit ihren Siedlern treten können.

- **Schnittstellen / Interface**

  Observable

- **Attribute**
  - **Entity[0…*] playersOnMe:** Speichert die Entitäten, die auf einem solchen Objekt stehen.

- **Methoden**
  - **getPosition():** Gibt die Position des Objektes zurück.
  - **String getName()**: Gibt den Namen des SteppableSpaceObjectes zurück, falls es existiert.
  - **boolean drillLayer()**: Falls es eine Asteroid ist, reduziert die Größe der Kortikalis des Asteroiden um eins. Übergibt die Bohraufgabe an das Layer.
  - **Ressource mineResource()**: Falls es eine Asteroid ist, baut es das Innere des Asteroiden ab und setzt die Art des Rohmaterials auf leer. Leitet die Mining-Aufgabe an das Core weiter.
  - **addResource(ressource: Resource)**: Übergibt die Aufgabe der Rohstoffzugabe an das Core, falls es eine solche Objekt ist, welches ein Core (Kern) hat.
  - **setMyPosition(newPosition: Position):** Setzt die Position eines Portals, falls solche Operation interpretierbar ist.
  - **boolean isActive():** Gibt an, ob ein Portal, oder Objekt aktiv ist, also ob es wirklich funktioniert.
  - **boolean setPair(pairGate: Gate):** Bindet das aktuelle Objekt mit einem Anderen zu.
  - **SteppableSpaceObject getPair():** Gibt das Paar eines Portals/SteppableSpaceObject zurück.

3.3.15 Sun

- **Verantwortung**

  Startet Sunflairs, was gefährlich für Siedler und Roboter sind.

- **Schnittstellen / Interface**

  EventObservable

- **Attribute**
  - **Position position:** Position der Sonne auf dem Spielfeld.

- **Methoden**
  - **doSunFlair():** Jede Entität auf dem Feld wird überprüft, ob sie in Gefahr ist, und wenn ja, stirbt sie.

3.3.16 Observable

- **Verantwortung**

  Diese Schnittstelle hat Operationen, dadurch Entitäten (Siedler und Roboter) zur Kollektion von SteppableSpaceObjects hinzugefügt werden können.

- **Methoden**
  - **checkOut(leavingEntity: Entity):** Das SteppableSpaceObject entfernt die Entität von sich.
  - **checkIn(newEntity: Entity):** Das SteppableSpaceObject registriert eine ankommende Entität an sich.

3.3.16 Observer

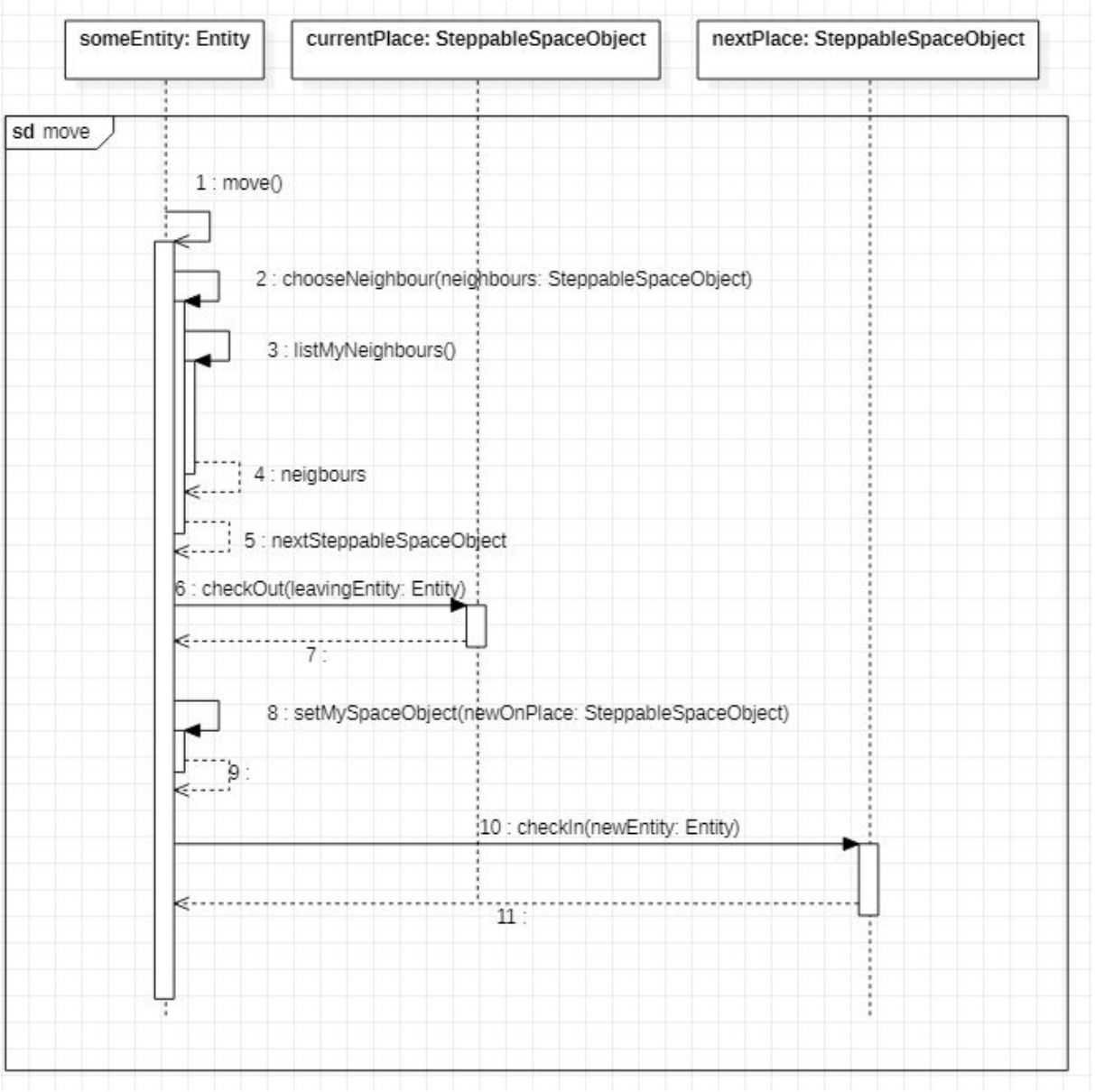
- **Verantwortung**

  Diese Schnittstelle hat das Aufgabe, dass die Entitäten auf SolarFlair, oder Asteroidexplosion reagieren. Diese Methoden der Interface hat die Kenntnisse, wie das aktuelle Entität das Ereignisse behandelt.

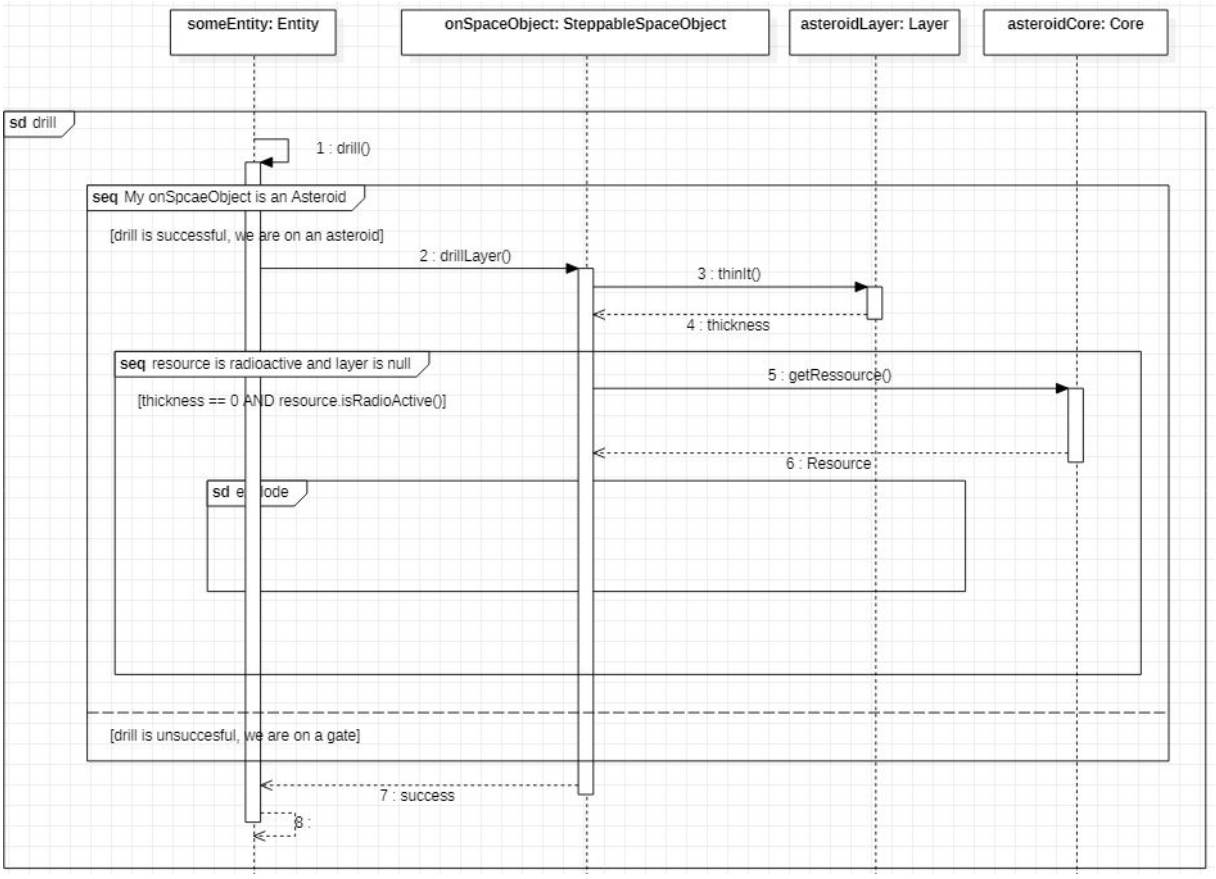
- **Methoden**
  - **notifyFlairEvent():** Benachrichtigt die Entität (und Spieler auch) über einem FlairEreignis, und reagiert darauf mit Methodenaufrufe der konkrete Objekt.
  - **notifyFlairDanger():** Benachrichtigt die Entität (und Spieler auch) über einem bevorstehende FlairEreignis, und reagiert darauf mit Methodenaufrufe der konkrete Objekt. Often nur ein Message auf dem Bildschirm, über die zukünftige Solarflair.
  - **notifyAsteroidExplosion():** Benachrichtigt die Entität (und Spieler auch), dass ihre Asteroid explodiert hat, und soll sie (die Entität) dieses Ereignis irgendwie abhängig von ihrem Typ behandeln.

3.4 Sequence Diagramme

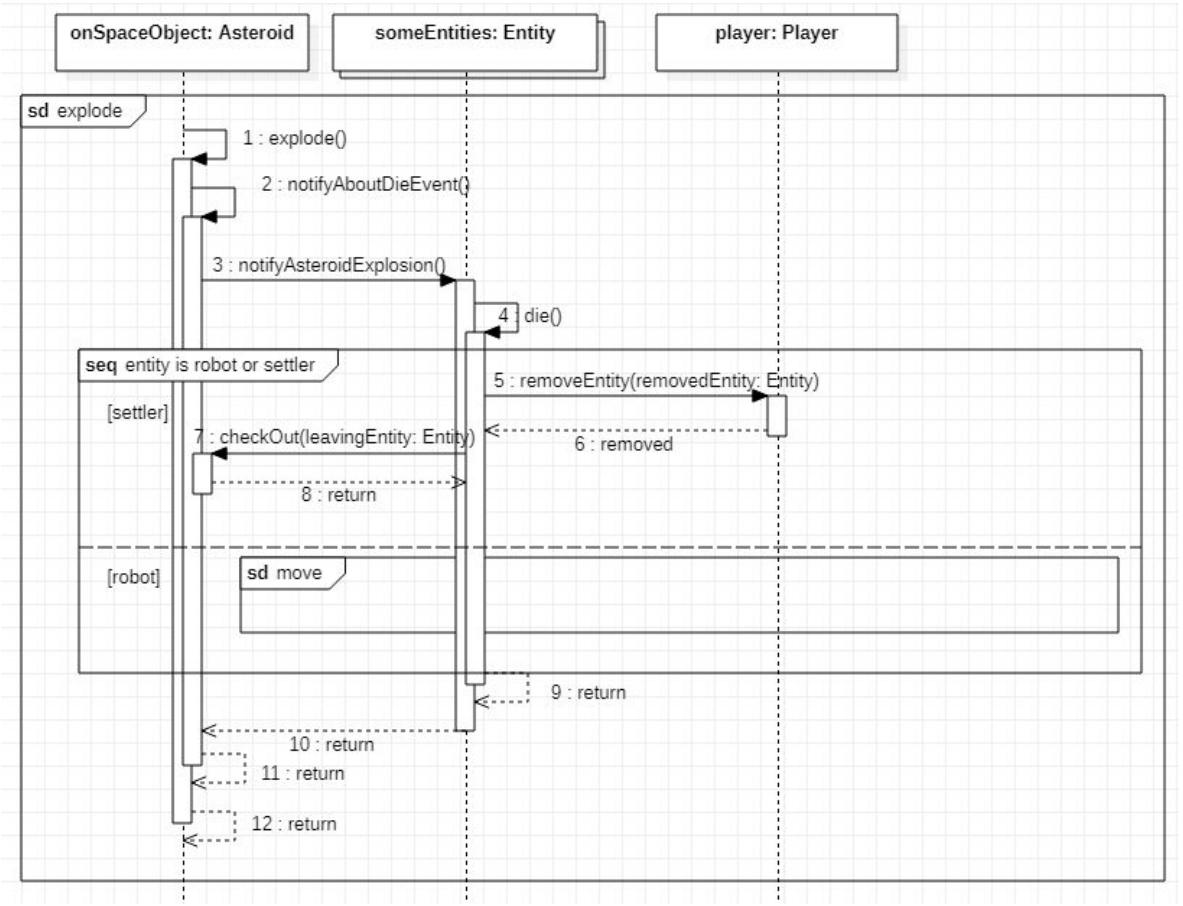
3.4.1 Move



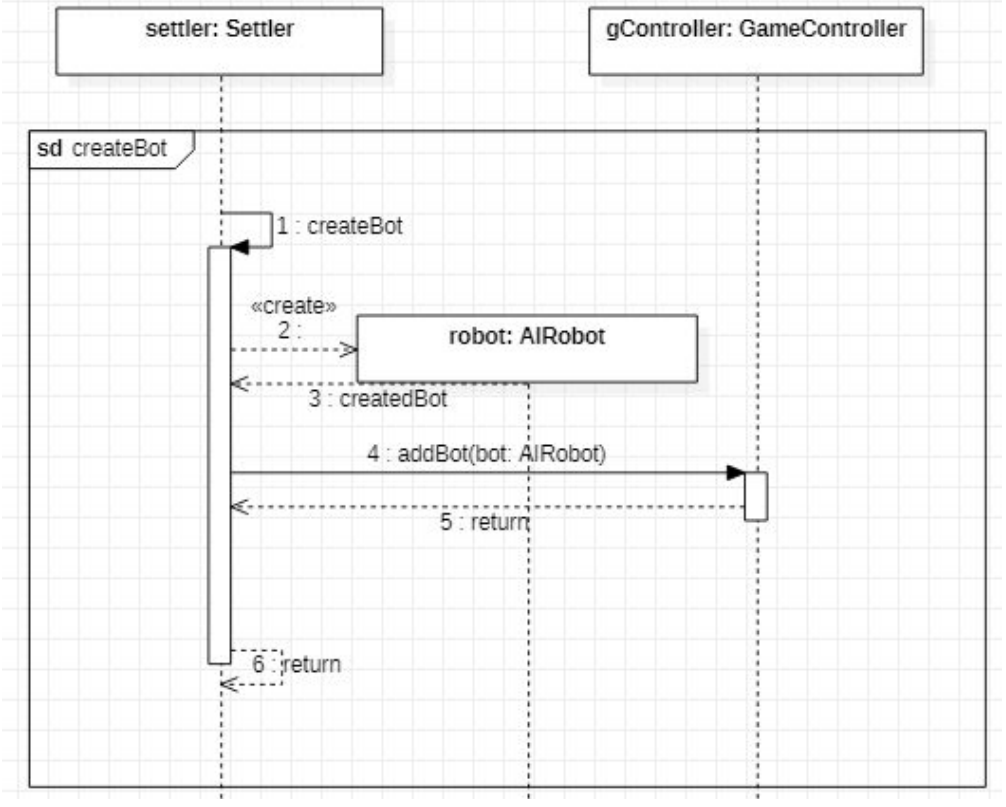
3.4.2 Drill


someEntity: Entity
onSpaceObject: SteppableSpaceObject
asteroidLayer: Layer
asteroidCore: Core
sd drill
1 : drill()
seq My onSpcaeObject is an Asteroid
[drill is successful, we are on an asteroid]
2 : drillLayer()
3 : thinIt()
4 : thickness
seq resource is radioactive and layer is null
5 : getRessource()
[thickness == 0 AND resource.isRadioActive()]
6 : Resource
sd explode
[drill is unsuccesful, we are on a gate]
7 : success
8 :

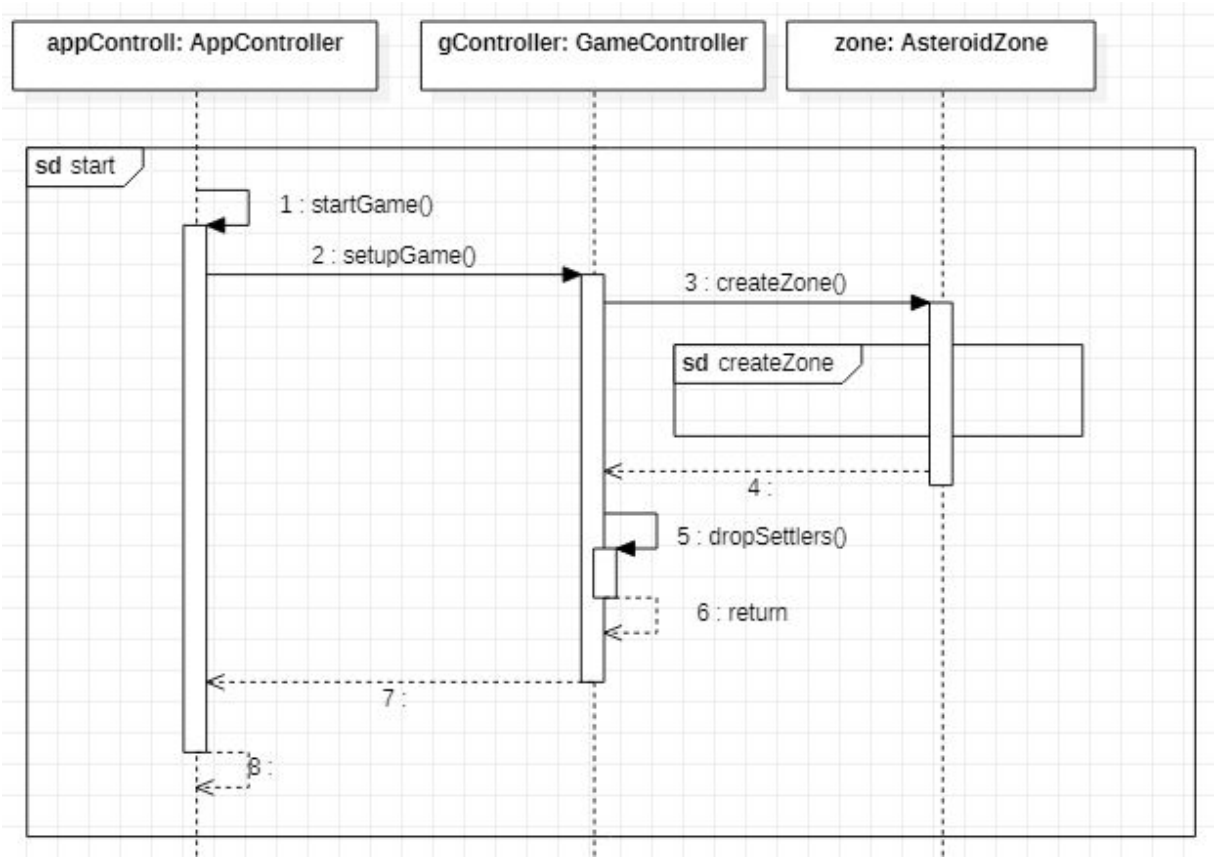
3.4.4 Explode



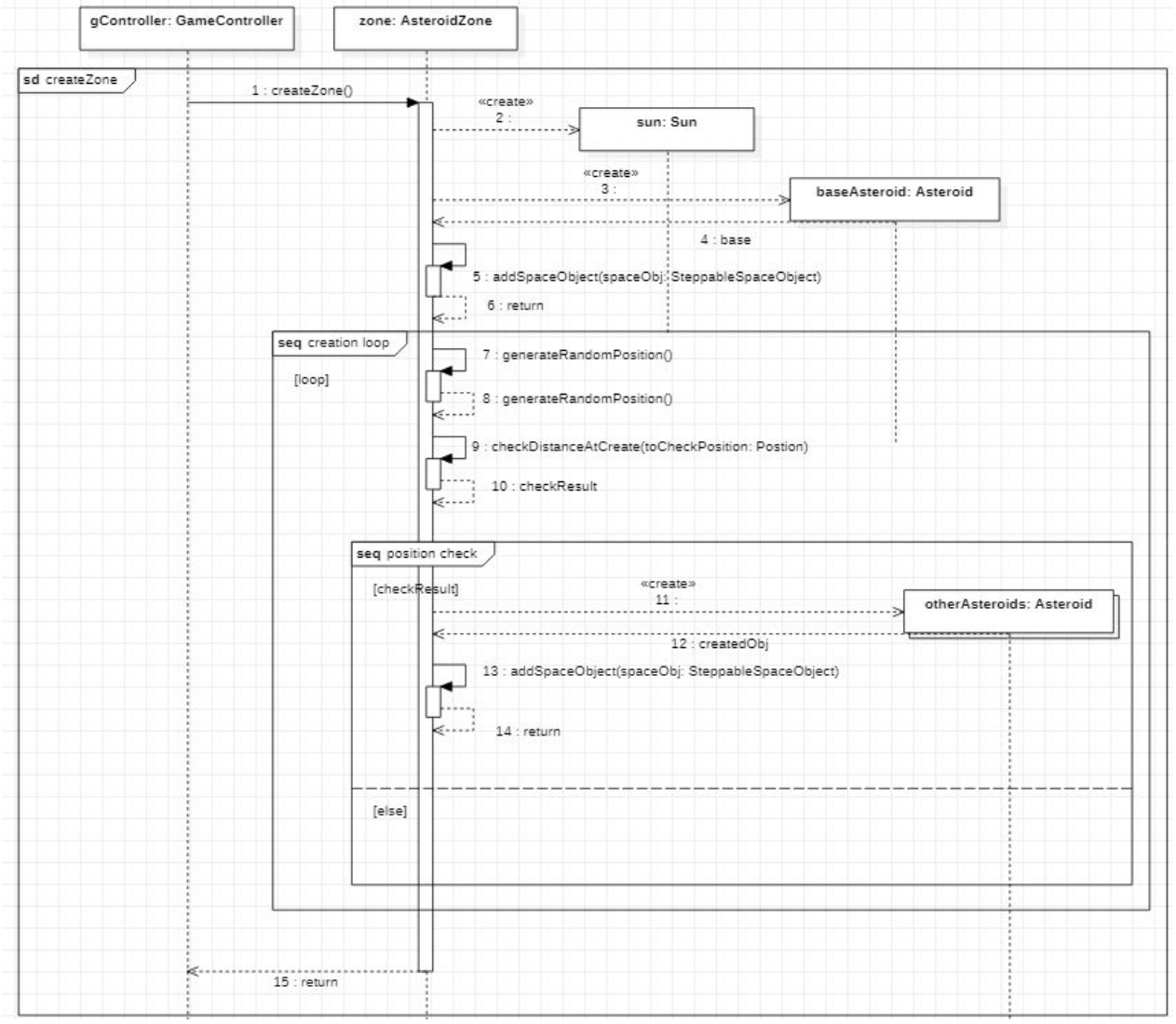
3.4.5 CreateBot



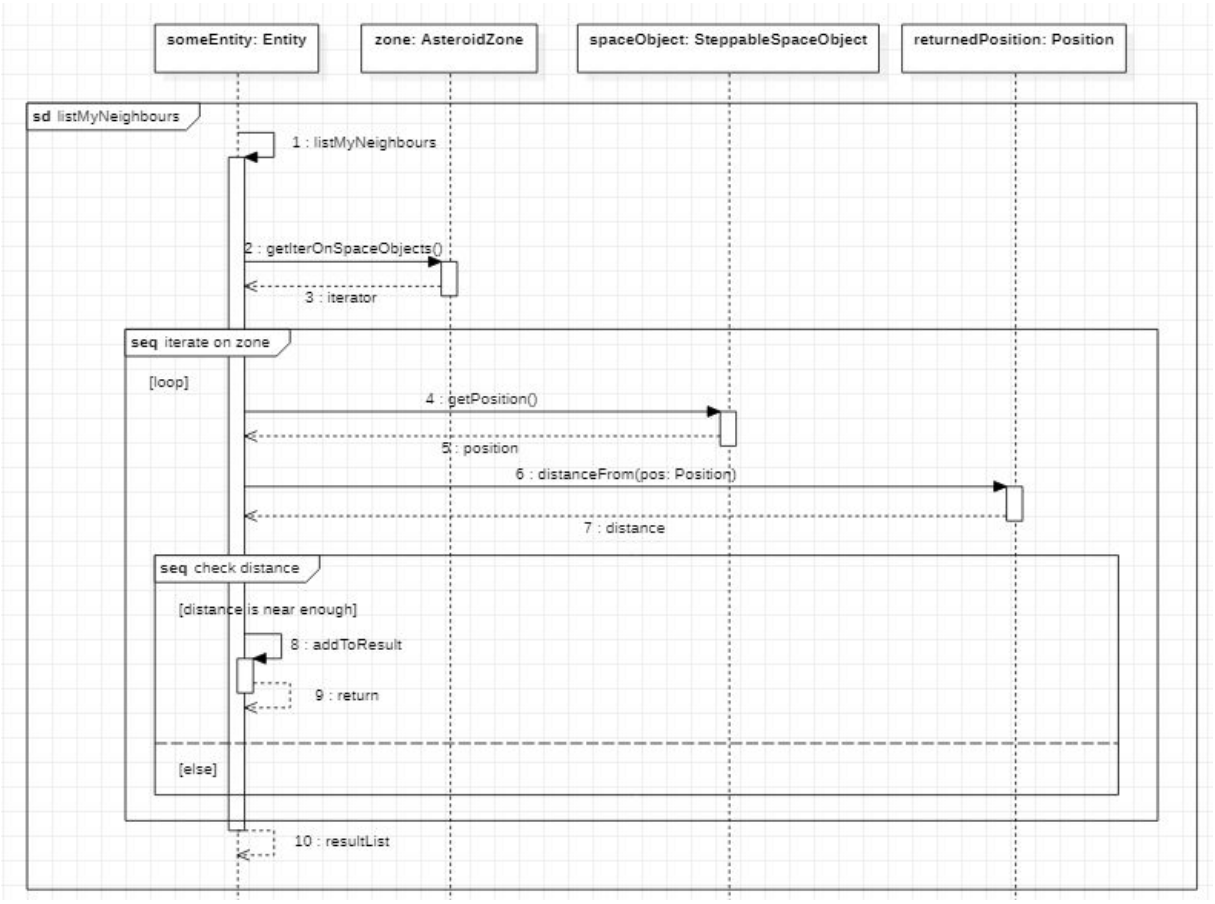
3.4.6 Start


appControll: AppController
gController: GameController
zone: AsteroidZone
sd start
1 : startGame()
2 : setupGame()
3 : createZone()
sd createZone
4 :
5 : dropSettlers()
6 : return
7 :
8 :

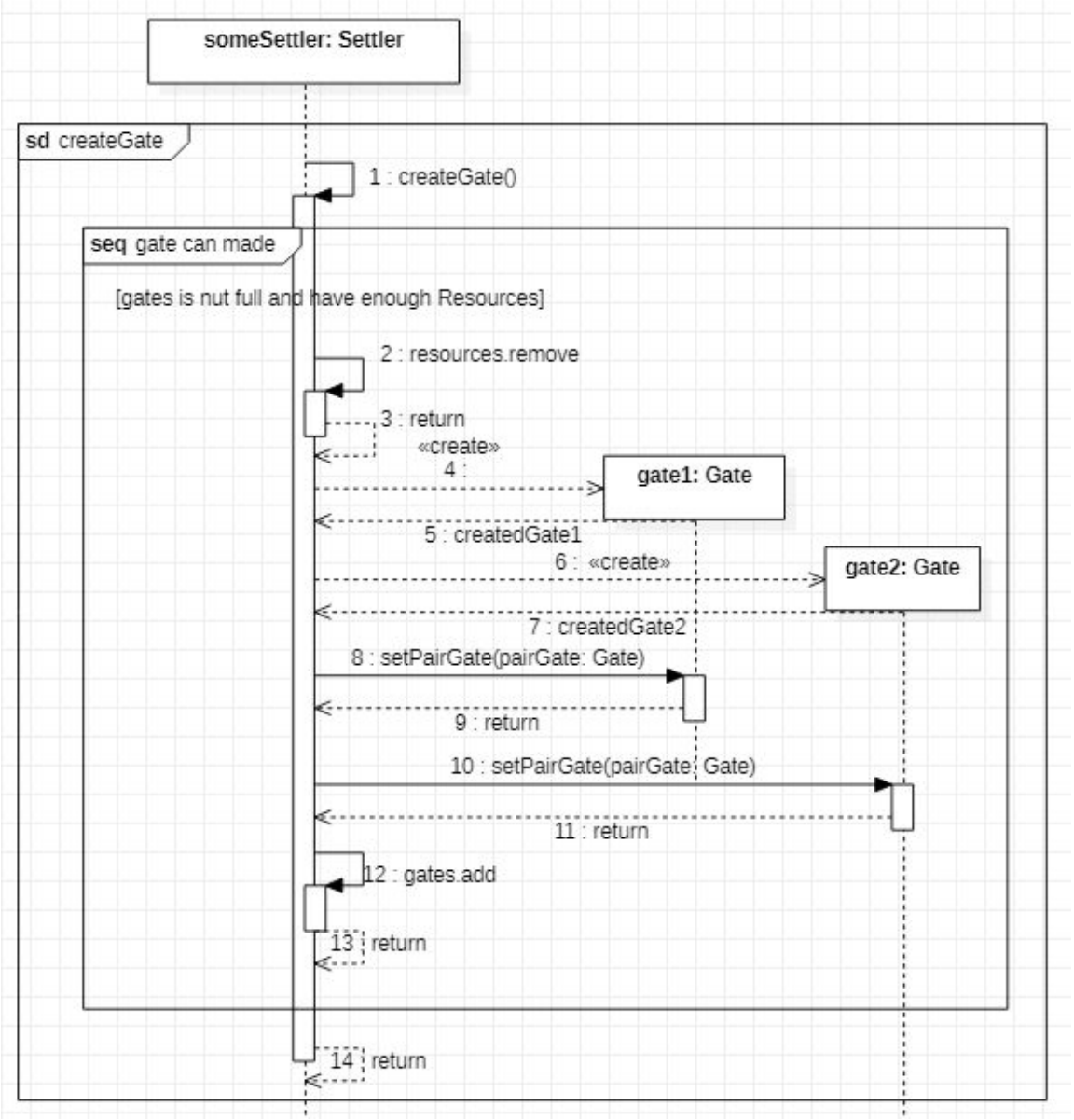
3.4.8 CreateZone



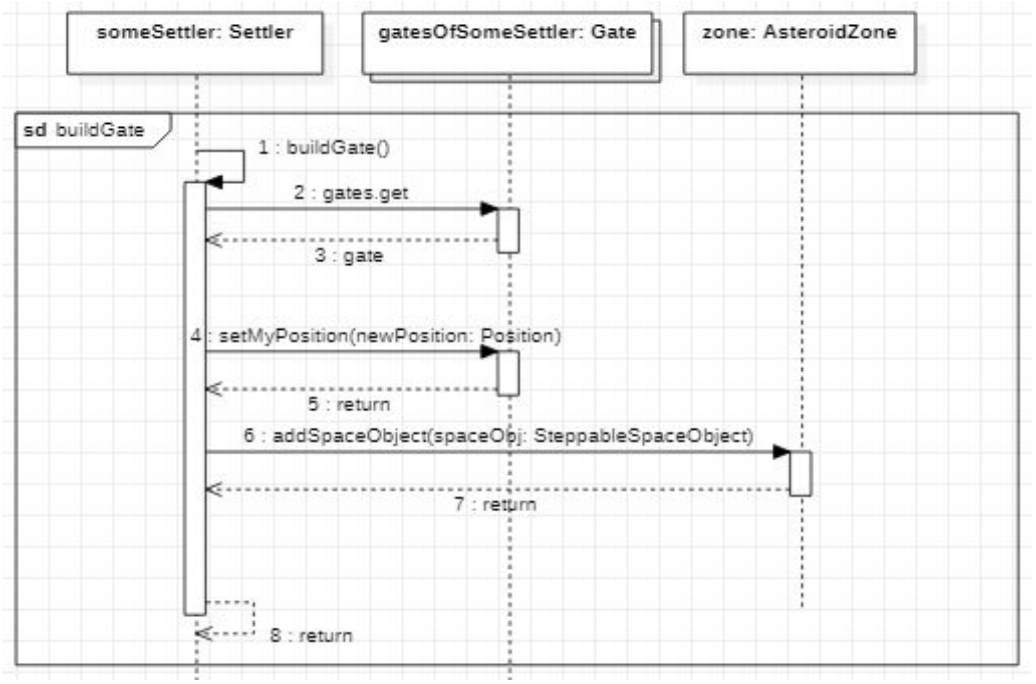
3.4.9 ListMyNeighbours



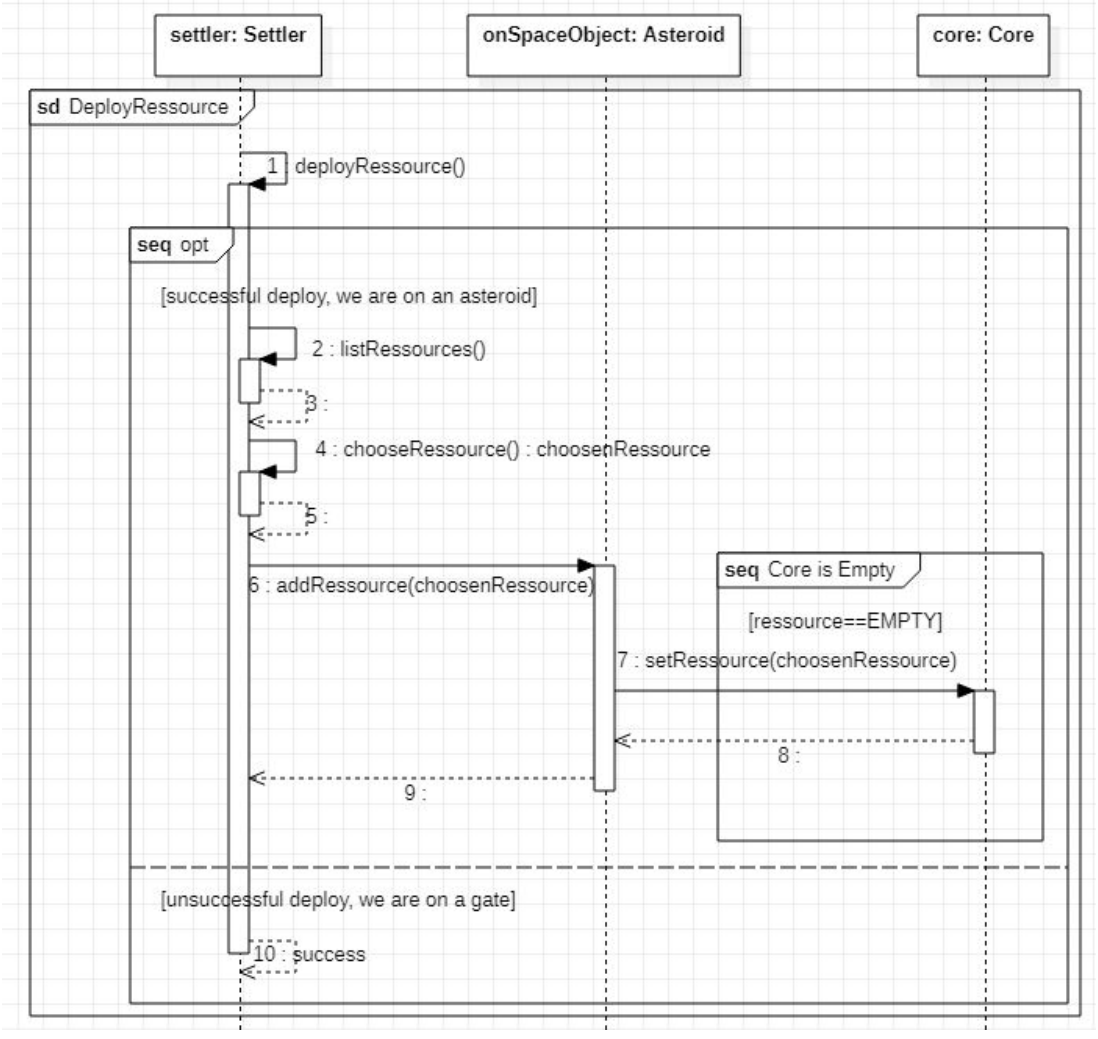
3.4.10 CreateGate



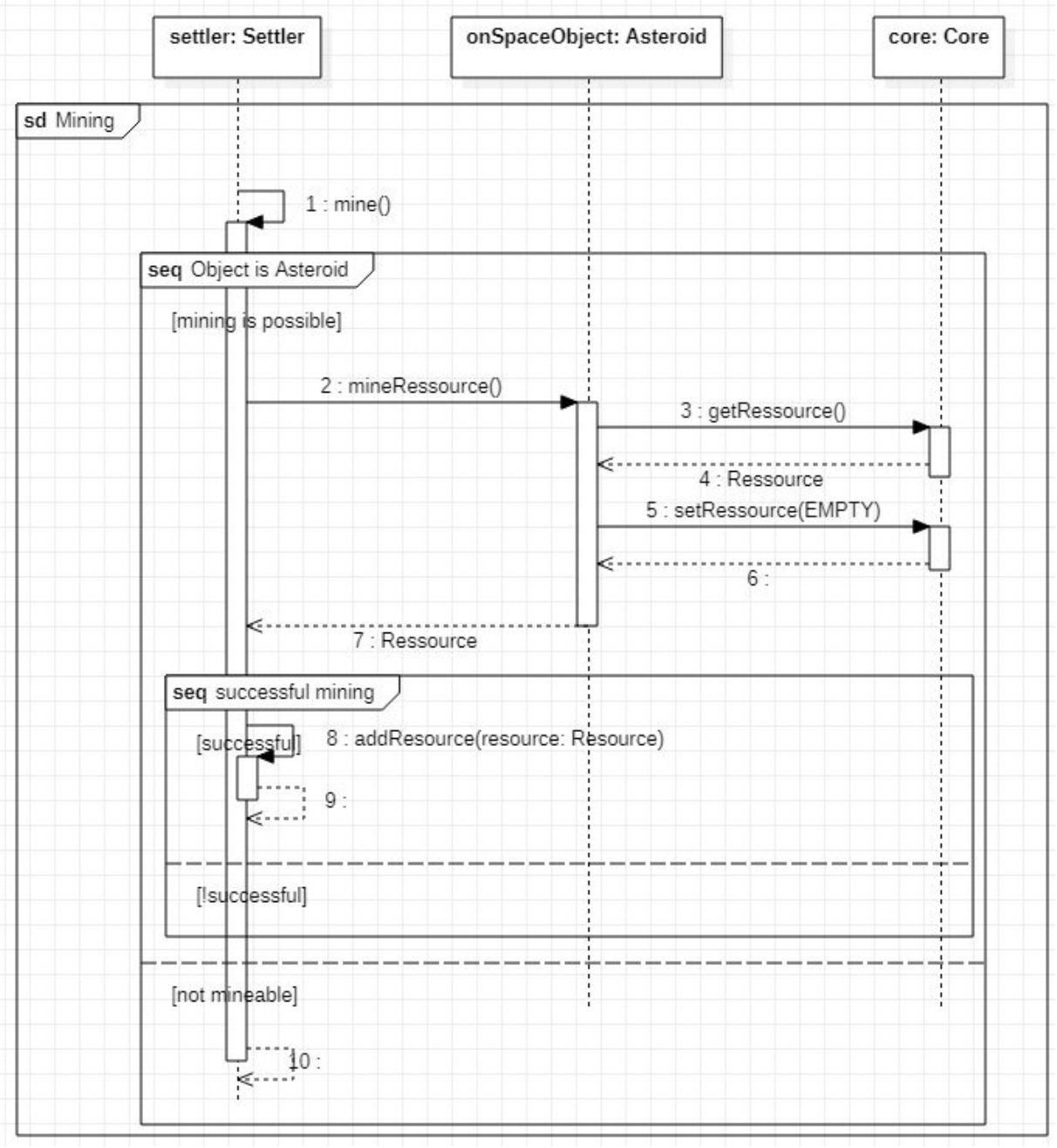
3.4.11 BuildGate



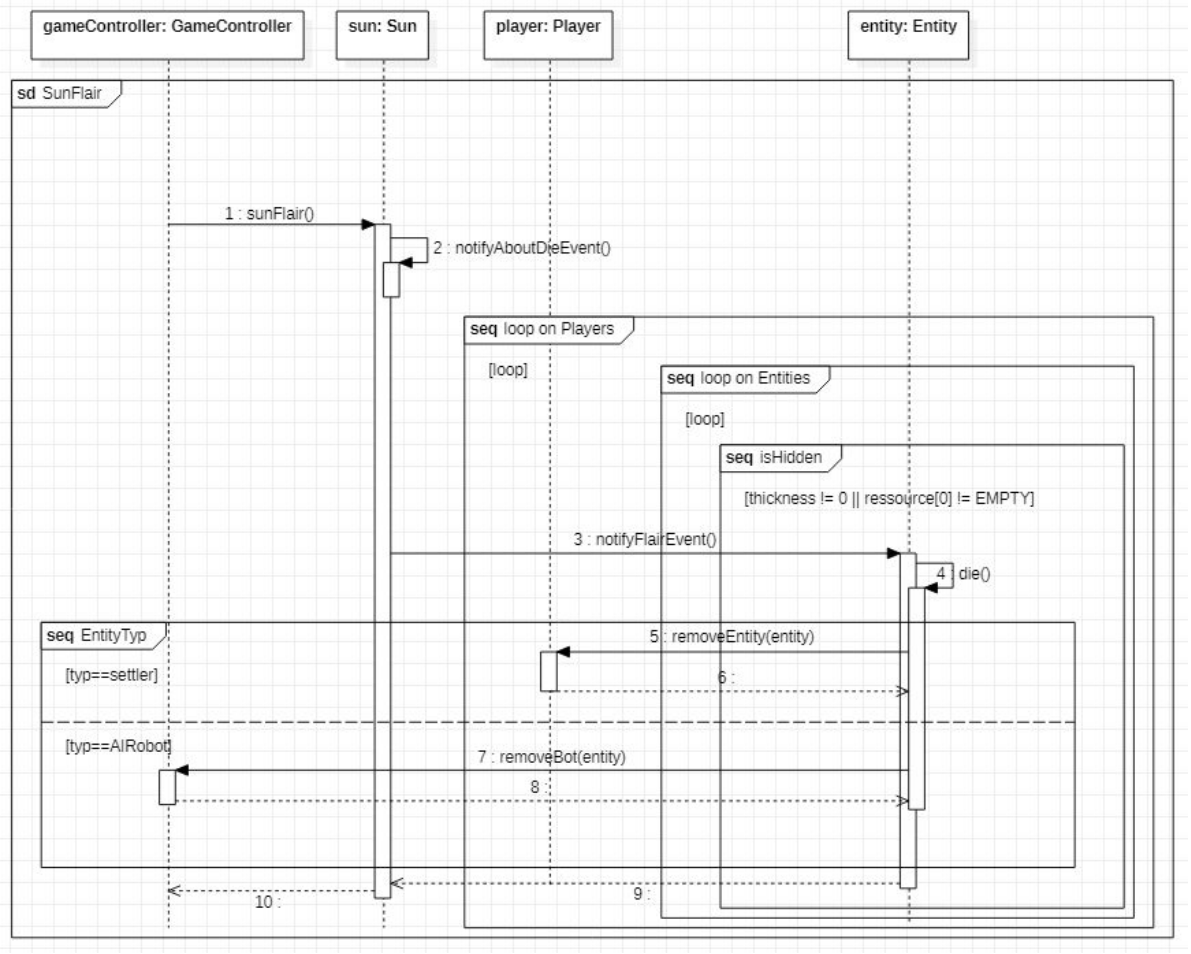
3.4.12 DeployResource



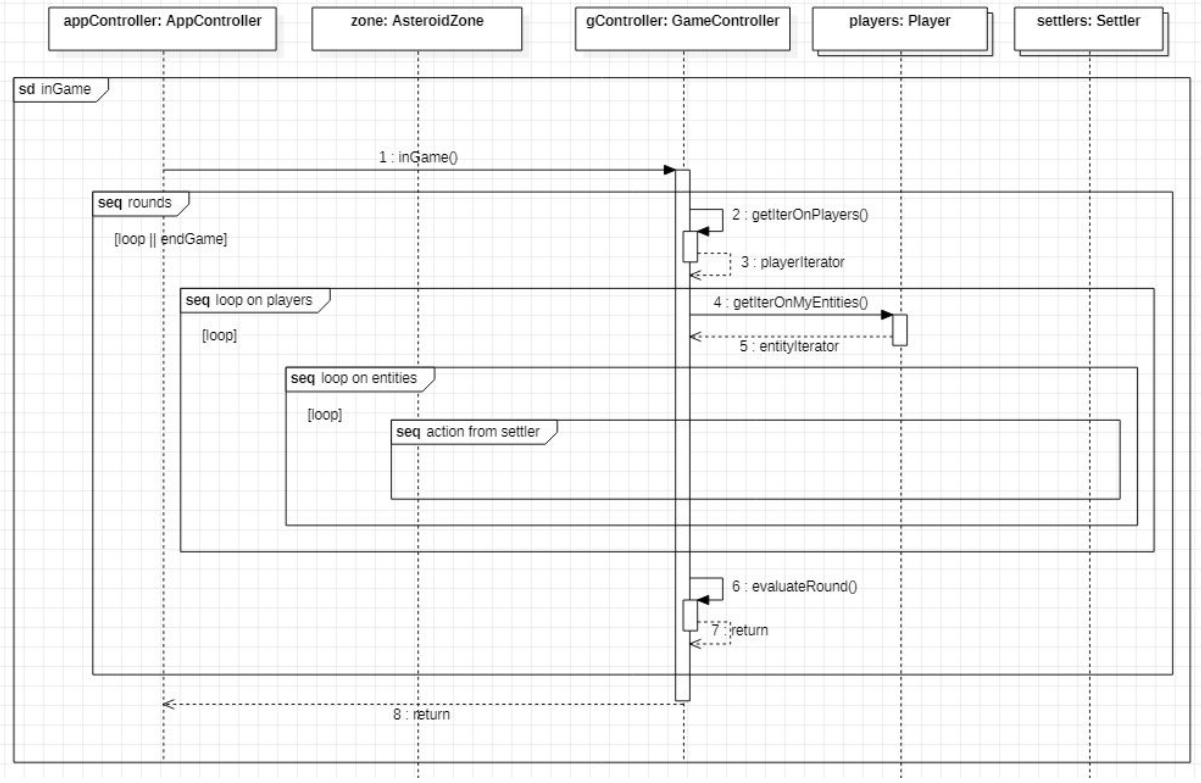
3.4.13 Mining


settler: Settler
onSpaceObject: Asteroid
core: Core
sd Mining
1 : mine()
seq Object is Asteroid
[mining is possible]
2 : mineRessource()
3 : getRessource()
4 : Ressource
5 : setRessource(EMPTY)
6 :
7 : Ressource
seq successful mining
[successful]
8 : addResource(resource: Resource)
9 :
[!successful]
[not mineable]
10 :

3.4.14 SunFlair

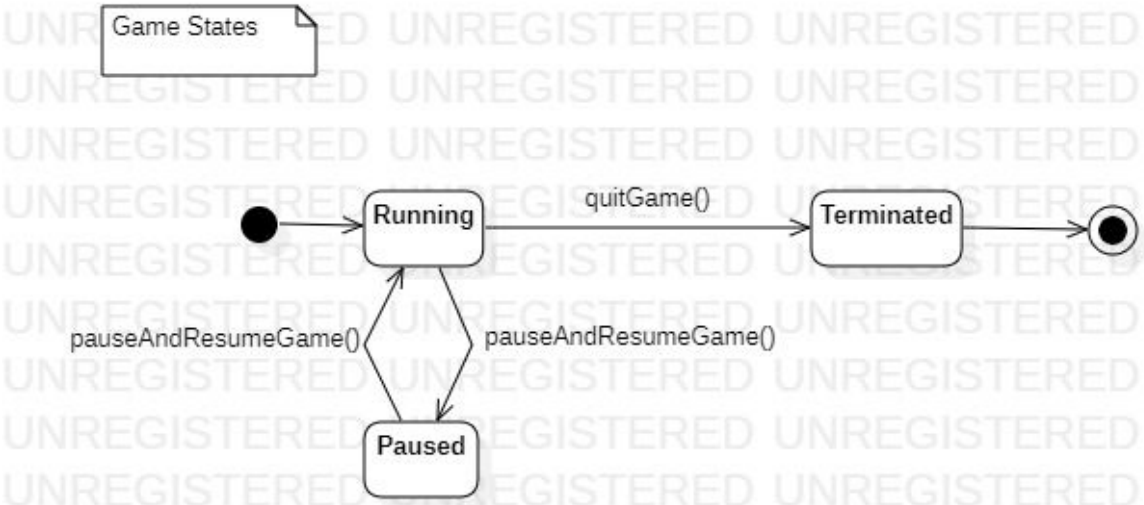


3.4.15 InGame

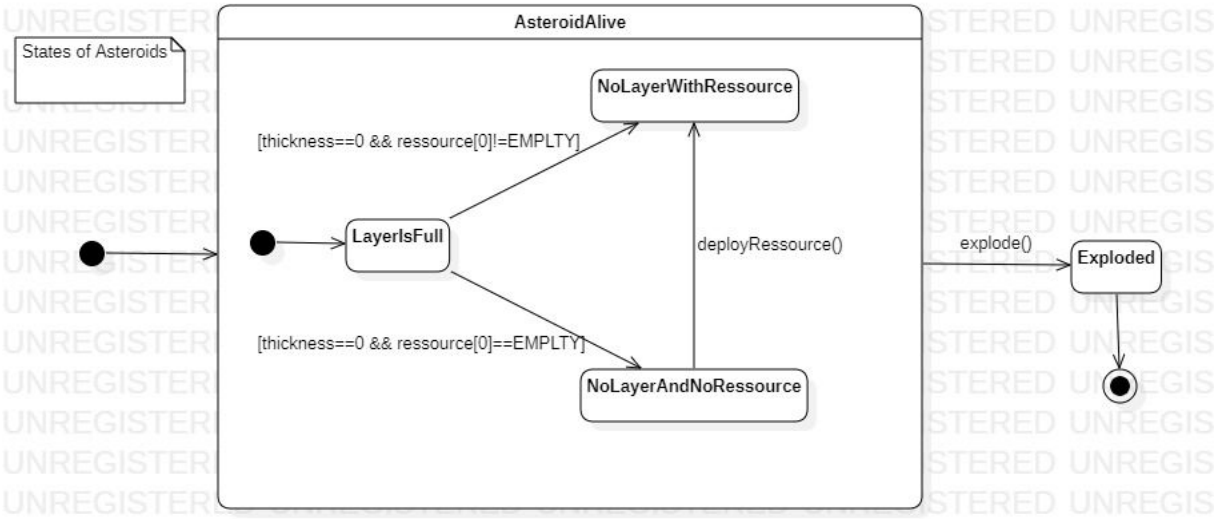


3.5 State-chart Diagramme

3.5.1 Game States



3.5.2 Asteroid States



3.6 Tagebuch

Anfang	Zeitaufwand	Teilnehmer	Beschreibung
2021.02.17. 15:00	2 Stude	Domokos Borbély Hrotkó Pongrácz	Meeting: Aufgabeausteilung auf diesem Etap, Jira eingeführt, über Klassendiagramm sprechen
2021.02.19. 13:00	2 Stunde	Domokos Borbély Hrotkó Pongrácz	Besprechen von Class Diagram
2021.02.20	2 Stunde	Pongrácz	Sequence Diagramme herstellen
2021.02.20	1 Stunde	Hrotkó	Auflistung und allg. Beschr. der Objekten
2021.02.21	2 Stunde	Pongrácz	Sequence Diagramme herstellen, kleine Änderungen in Klassendiagramm
2021.02.21	1,5 Stunde	Hrotkó	Sequenzdiagramme herstellen
2021.02.22.	1 Stunde	Borbély	Beschreibung der Klassen
2021.02.22.	1 Stunde	Domokos	Beschreibung der Klassen
2021.02.23.	1 Stunde	Domokos	Beschreibung der Klassen
2021.02.23.	1 Stunde	Pongrácz	Kleinere Änderungen, Klassenbeschreibung
2021.02.23.	1 Stunde	Borbély	Beschreibung der Klassen
2021.02.23	1 Stunde	Hrotkó	State Chart diagramme hestellen, durchlesen und kontrollieren das ganze Dokument.
2021.02.23	0,5 Stunde	Hrotkó Pongrácz Borbély Domokos	Aufgabenteil abgeschlossen, Diskussion
2021.02.25	1 Stunde	Pongrácz	Kleinere Änderungen