4. Ausarbeitung des Analysenmodells 2

31 – marci_und_die_mitbewohner

Konsulent:
Kovács Márton

Mitglieder

Seben Domonkos András	(ETBCNP)	domi.seben@gmail.com
Szapula László	(DJQOM9)	szapula.laszlo.99@gmail.com
Filip Krisztina	(QE4L0M)	fkriszta997@gmail.com
Golej Márton Marcell	(V1BYVS)	golejmarci@gmail.com
Visy Tamás	(CTSJ3H)	tamas.visy@gmail.com

4. Ausarbeitung des Analysemodells 1

4.1 Objektkatalog

4.1.1 **Controller**

Kontrolliert den Ablauf des Spiels, sorgt für die wichtigere, weniger spezifische Aufgaben. Solche Aufgaben sind zum Beispiel die Behandlung der Schneestürme, die Bewegung der Charaktere, und die Steuerung ihrer Arbeit (also die Tritte). Bewartet den Zusammenhang der Eisschollen und Charakters. Weiß von den Dingen, die in den Eisschollen gefroren sind.

4.1.2 **Eisscholle**

Die elementare Einheit des Eisfeldes. Diese Bausteine repräsentieren den Zustand und ihre Änderung von den kleineren Einheiten des Eisfeldes. Beispiel dafür kann die Speicherung der Schneeschichten, der Aufbau oder Verfallen von Iglus, der Stand der Eisschollen, (gut/umgedreht) beziehungsweise ihre Arten (stabil/instabil/Loch) und Tragfähigkeit sein.

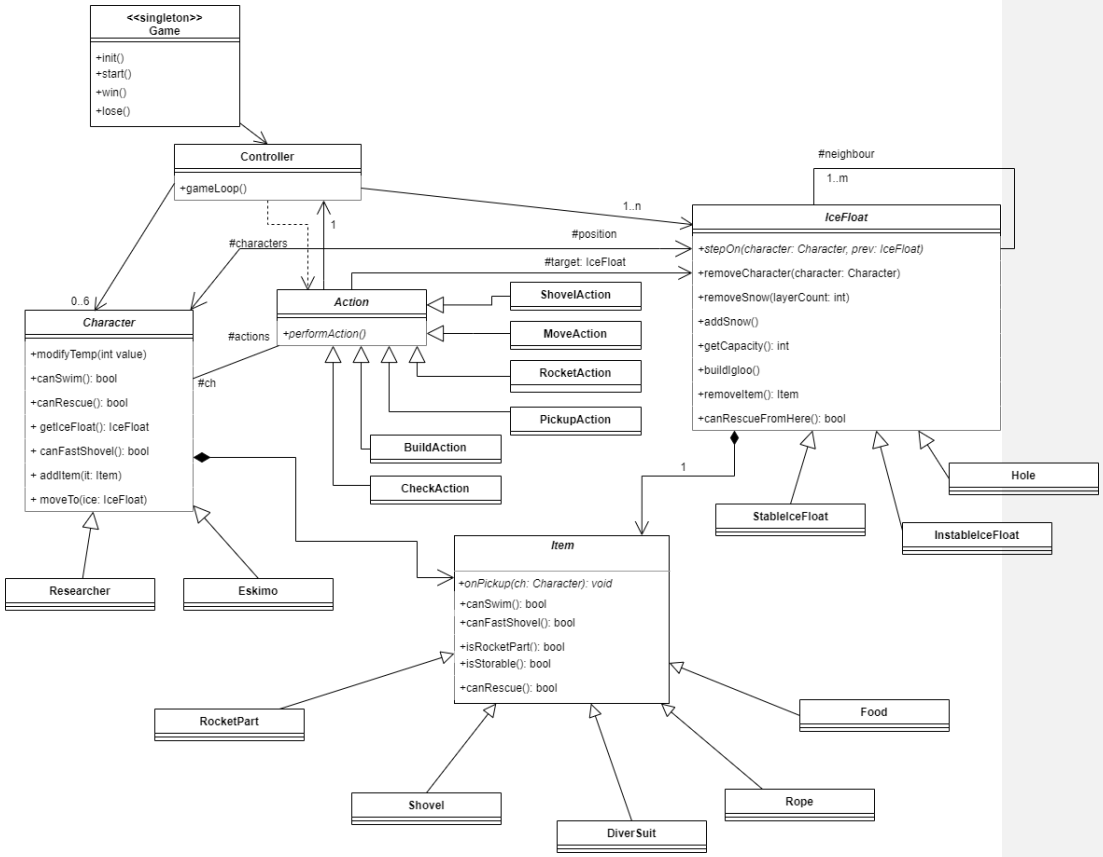
4.1.3 **Character (Spieler)**

Repräsentiert die Spieler. Es wird zwei Typen geben, ein Eskimo und ein Forscher. Ihr Ziel ist die Wartung der Körpertemperatur mit Essen und Arbeit. Besitzt und benutzt Gegenstände beziehungsweise schafft sie an.

4.1.4 **Gegenstand**

Die Gegenstände, die in den Eisschollen gefroren sind. Die Spieler schaffen sie an und verlagern sie sie. Es gibt 7 Typen davon: Schaufel, Seil, Taucheranzug, Lebensmittel und die 3 Bauteile der Leuchtpistole.

4.2 Statische Strukturdiagramme



4.3 Beschreibungen der Klassen

4.3.1 Game

- **Verantwortung**

Das Spiel zu initialisieren und starten.

- **Attribute**
  - **- controller**: Referenz zur Klasse “Controller”

- **Methoden**
  - **+ void init():** Diese Funktion initialisiert das Spiel. Also herstellt sie alle benötigte Klassen.
  - **+ void start():** Diese Funktion startet das Spiel. Es ruft also die Funktion “void gameLoop()” von der Klasse “Controller” an.
  - **+ void lose():** Wegen des Funktionsaufrufes wird das Spiel beendet und bekannt gegeben, dass die Spieler verloren haben.
  - **+void win():** Wegen des Funktionsaufrufes wird das Spiel beendet und bekannt gegeben, dass die Spieler gewonnen haben.

4.3.2 Controller

- **Verantwortung**

Das nacheinander Kommen von Spielern zu organisieren. Beim Wechseln zwischen Spieler ist diese Klasse auch verantwortlich für die Schneestürme.

- **Attribute**
  - **- icefloats**: Referenzlist zu den Eisschollen.
  - **- characters**: Referenzlist zu den Spielern.

- **Methoden**
  - **+ void gameLoop():** Diese Funktion ist verantwortlich für das nacheinander Kommen von Spielern, und beim Wechseln zwischen Spieler ruft die Funktion “*snowStorm()*” an.
  - **- void snowStorm():** Diese Funktion wählt einige Schneeschollen aus, und liegt Schnee darauf. Falls ein Spieler auf einer solchen Scholle steht, dann verliert er eine Einheit von Körpertemperatur außer des Falles, wenn dort ein Iglu ist. In diesem Fall verliert der Spieler keine Körpertemperatur, aber der Iglu wird zerstört.

4.3.3 Character

- **Verantwortung**

Die Enthaltung von den möglichen Arbeiten. Die Klasse ermöglicht die Nachverfolgung und Änderung der Körpertemperatur des Characters. Diese Klasse enthält auch die Gegenstände, die der Spieler aufgenommen hat, und hat einige Funktionen um zu bestimmen, ob der Charakter dazu fähig ist, bestimmte Arbeiten durchzuführen.

- **Attribute**
  - **#position**: Die Eisscholle, auf der der Character sich befindet.
  - **#items**: Referenzlist auf den Gegenständen, die der Character besitzt.
  - **#actions:** Eine Collection von Aktionen, die der Character benutzen kann.
  - **-temp:** Repräsentiert die Körpertemperatur von dem Character.

- **Methoden**
  - **+void modifyTemp(value: int)**: Modifiziert die Körpertemperatur des Characters um "value". Das wird typischerweise beim Schneesturm und bei der Aufnahme von Essen angerufen.
  - **+bool canSwim():** Gibt bekannt, ob der Character in einem Loch schwimmen kann.
  - **+bool canRescue():** Gibt bekannt, ob der Spieler andere Spieler retten kann, die in einem Loch gefallen sind.
  - **+IceFloat getIceFloat():** Gibt den Eisscholle zurück, auf dem der Character steht.
  - **+bool canFastShovel():** Gibt bekannt, ob der Spieler einen Schaufel hat.
  - **+void addItem(it: Item)**: Durch diese Funktion können Gegenstände zum Lager des Charakters hinzugefügt werden.
  - **+void moveTo(ice: IceFloat)**: Überliefert den Character zum Eisscholle.

4.3.4 Researcher

- **Verantwortung**

Einen solchen Character zu realisieren, der herausfinden kann, wie viele Characters die benachbarten Eisschollen ertragen können.

- **Basisklasse**

Character

4.3.5 Eskimo

- **Verantwortung**

Einen solchen Character zu realisieren, der Iglus aufbauen kann.

- **Basisklassen**

Character

4.3.6 Item

- **Verantwortung**

Das soll für einen abstrakten Gegenstand entsprechen, das die verschiedene Eigenschaften der einzelne Gegenstand beschreibt.

- **Methoden**
  - **+abstract void onPickup(ch: Character):** Sagt, was passiert, wenn ein Character dieses Gegenstand aufnimmt.
  - **+bool canSwim():** Sagt, ob der Character, der dieser Gegenstand hat, aus einen Loch schwimmen kann.
  - **+bool canFastShovel():** Sagt, ob der Character mehr Schnee mit einem Aktion schaufeln kann.
  - **+bool isRocketPart():** Sagt, ob der Gegenstand ein Teil der Raket ist.
  - **+bool isStoreable():** Sagt, ob der Gegenstand in den Lager des Spielers gesteckt werden kann.
  - **+bool canRescue():** Sagt, ob der Character einen anderen Character retten kann, der in eine Loch eingefallen ist.

4.3.7 RocketPart

- **Verantwortung**

Die Bestandteile der Leuchtpistole zu identifizieren und es sagen können, ob die Bestandteile aufgenommen wurden.

- **Basisklasse**

Item

- **Methoden**
  - **+abstract void onPickup(ch: Character):** Sagt, was passiert, wenn ein Character dieses Gegenstand aufnimmt.

4.3.8 DiverSuit

- **Verantwortung**

Es soll vermöglichen, dass ein Spieler überlebt einen Fall in ein Loch.

- **Basisklasse**

Item

- **Methoden**
  - **+void onPickup(ch: Character):** Sagt, was passiert, wenn ein Character dieses Gegenstand aufnimmt.

4.3.9 Shovel

- **Verantwortung**

Falls ein Character diesen Gegenstand enthält, dann kann er 2 Schnee mit einem Action schaufeln.

- **Basisklasse**

Item

- **Methoden**
  - **+abstract void onPickup(ch: Character):** Sagt, was passiert, wenn ein Character dieses Gegenstand aufnimmt.

4.3.10 Rope

- **Verantwortung**

Falls ein Character diesen Gegenstand enthält, dann kann er einen anderen Character aus einem benachbarten Lochfeld retten.

- **Basisklasse**

Item

- **Methoden**
  - **+abstract void onPickup(ch: Character):** Sagt, was passiert, wenn ein Character dieses Gegenstand aufnimmt.

4.3.11 Food

- **Verantwortung**

Beim Aufnehmen dieses Gegenstandes bekommt man +1 Temperatur.

- **Basisklasse**

Item

- **Methoden**
  - **+abstract void onPickup(ch: Character):** Sagt, was passiert, wenn ein Character dieses Gegenstand aufnimmt.

4.3.12 IceFloat

- **Verantwortung**

Das ist eine abstrakte Klasse, das irgendwie regulieren muss, wie die Spieler auf sie treten. Ansonsten ist sie verantwortlich für die Schneeschichten und die Iglus, die sie charakterisieren.

- **Attribute**
  - **#neighbour:** Eine Sammlung der benachbarten Eisschollen
  - **#item:** der Gegenstand, der in der Eisscholle gefroren ist (der gibt es nicht immer)
  - **#characters:** Die Characters, die auf dieser Eisscholle stehen.
  - **#iglu:** Durch diesem Attribut kann man sagen, ob es auf der Eisscholle ein Iglu gibt.
  - **#snowLevel:** Repräsentiert die Menge von Schnee auf der Eisscholle.
  - **#capacity:** Durch diesem Attribut kann man beobachten, wie viel Character der Eisscholle fördern kann.
- **Methoden**
  - **+abstract boolean stepOn(Character ch, IceFloat prev)**: Die Methode behandelt die Anfrage eines Characters, der auf diese Eisscholle treten möchte.
  - **+void removeCharacter(Character ch)**: Das entfernt der Character von einer Eisscholle.
  - **+void removeSnow(int layerCount)**: Die bestimmte Anzahl von Schichten werden von der Eisscholle entfernt.
  - **+void addSnow()**: Eine Schicht Schnee wird auf die Eisscholle gelegt. Sie wird von dem Schneesturm gerufen.
  - **+int getCapacity()**: Die Methode gibt ihre Kapazität bekannt.
  - **+void buildIgloo()**: Die Methode erschafft ein Iglu.
  - **+Item removeItem()**: Durch dieser Methode wird das Gegenstand von der Eisscholle zu dem Charakter übernommen.
  - **+bool canRescueFromHere():** Gibt es zurück, ob ein Character von einem Loch mit einem Seil aufgezogen werden kann.

4.3.13 StableIceFloat

- **Verantwortung**

Die allgemeine Eisscholle, der alle Characters halten kann, ohne umzukippen.

- **Basisklasse**

IceFloat

- **Methoden**
  - **+bool stepOn(Character character, IceFloat prev)**: Das akzeptiert den Character auf diese Eisscholle

4.3.14 InstableIceFloat

- **Verantwortung**

Nach einem bestimmten Anzahl von Character kippt es um.

- **Basisklasse**

IceFloat

- **Methoden**
  - **+bool stepOn(Character character, IceFloat prev)**: Das akzeptiert den Character auf diese Eisscholle

4.3.15 Hole

- **Verantwortung**

Falls man darauf tritt, fällt er ins Wasser.

- **Basisklasse**

IceFloat

- **Methoden**
  - **+bool stepOn(Character character, IceFloat prev)**: Das akzeptiert den Character auf diese Eisscholle

4.3.16 Action

- **Verantwortung**

Abstrakte Klasse, ihre Kinder realisieren die einzelne verschiedene Aktionen (Arbeiten) der Spieler.

- **Attribute**
  - **- controller:** Bietet Zugriff zum Kontroller, z.B. um
  - **# ch**: Speichert ein Charakter, zu wem die gegebene Aktion gehört.
  - # **target**: Wo die gegebene Aktion durchgeführt wird.
- **Methoden**
  - **+ void performAction():** Durchführt den Action.

4.3.17 ShovelAction

- **Verantwortung**

Schaufelt Schnee auf dem gegebenen Eisscholle.

- **Basisklasse**
  - **Action**
- **Methoden**
  - **+ void performAction():** Durchführt den Action.

4.3.18 MoveAction

- **Verantwortung**

Bewegt den Character auf einen anderen Eisscholle.

- **Basisklasse**
  - **Action**
- **Methoden**
  - **+ void performAction():** Durchführt den Action.

4.3.19 RocketAction

- **Verantwortung**

Baut die Rakete zusammen, falls alle Teile auf einem Scholle sind.

- **Basisklasse**
  - **Action**
- **Methoden**
  - **+ void performAction():** Durchführt den Action.

4.3.20 PickupAction

- **Verantwortung**

Nimmt den Gegenstand von dem gegebenen Eisscholle auf.

- **Basisklasse**
  - **Action**
- **Methoden**
  - **+ void performAction():** Durchführt den Action.

4.3.21 BuildAction

- **Verantwortung**

Baut einen Iglu auf den gegebenen Eisscholle.

- **Basisklasse**
  - **Action**
- **Methoden**
  - **+ void performAction():** Durchführt den Action.

4.3.22 CheckAction

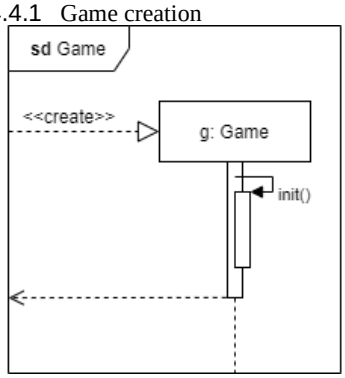
- **Verantwortung**

Überprüft den Kapazität einen Eisscholle.

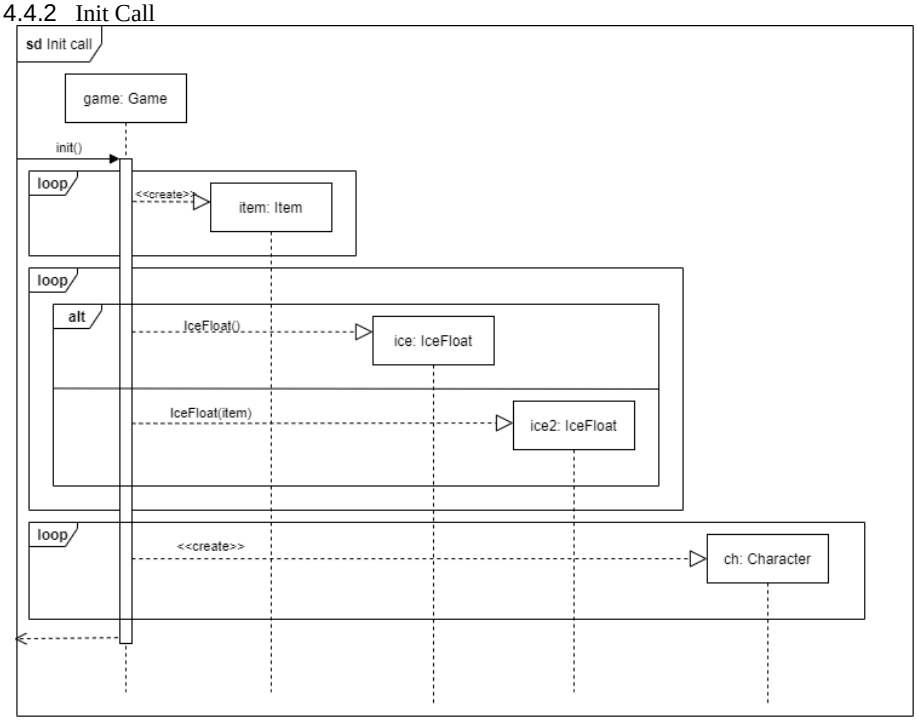
- **Basisklasse**
  - **Action**
- **Methoden**
  - **+ void performAction():** Durchführt den Action.

4.4 Sequenzdiagramme

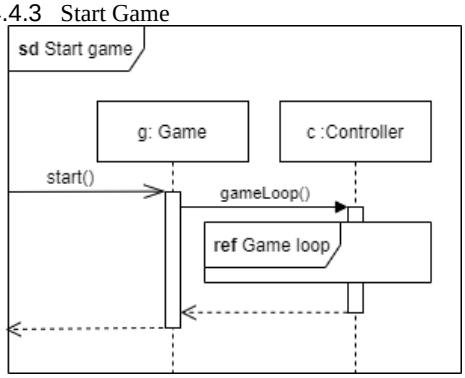
4.4.1 Game creation



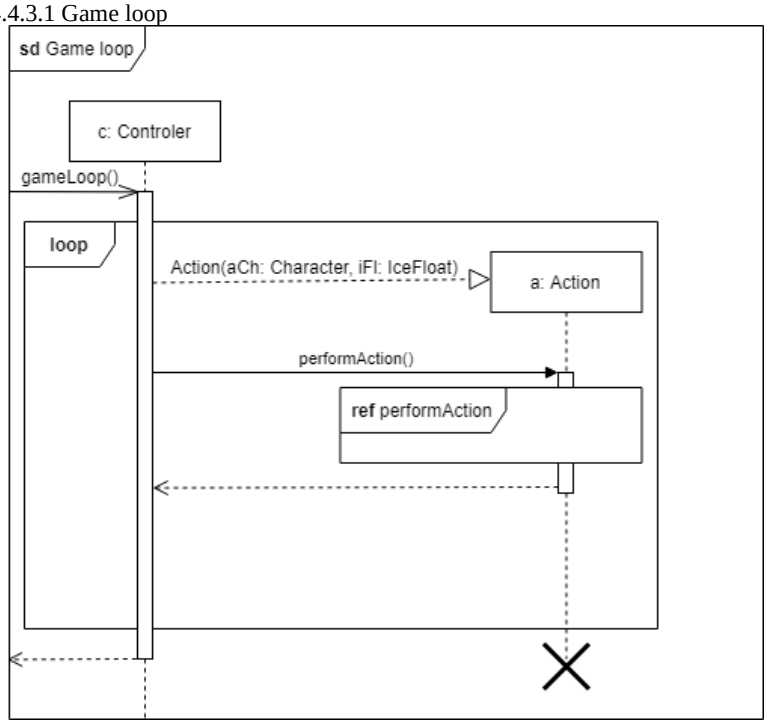
4.4.2 Init Call



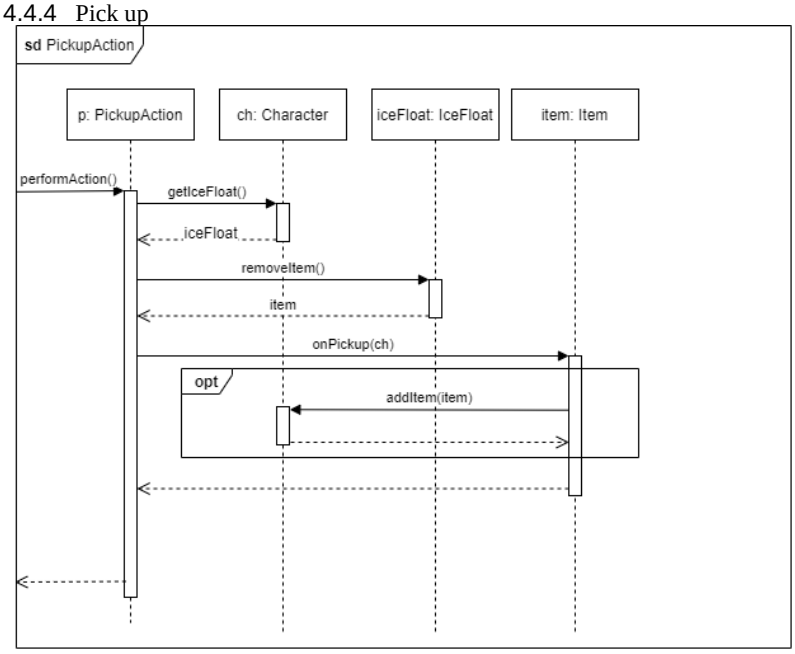
4.4.3 Start Game



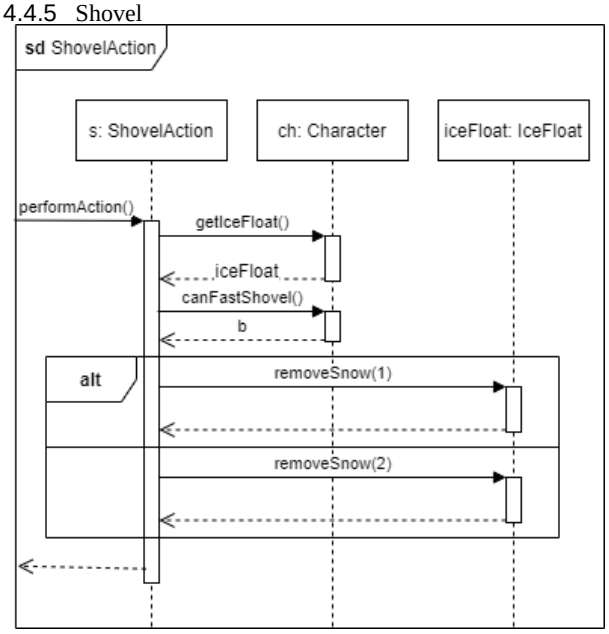
4.4.3.1 Game loop



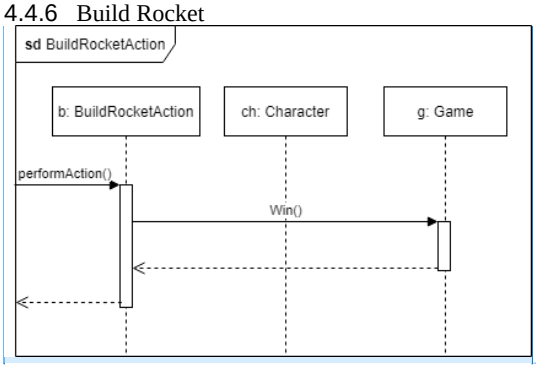
4.4.4 Pick up



4.4.5 Shovel

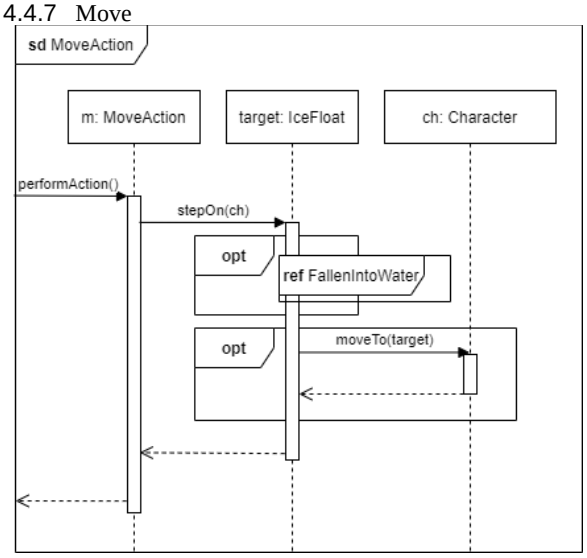


4.4.6 Build Rocket

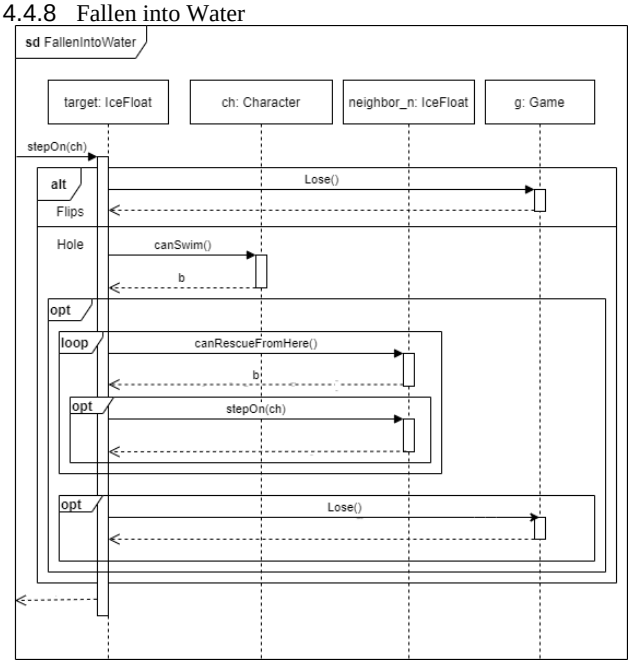


Commented [1]: hogyan adunk hozzá actionöket a Characterhez? kéne vmi addAction, meg removeAction

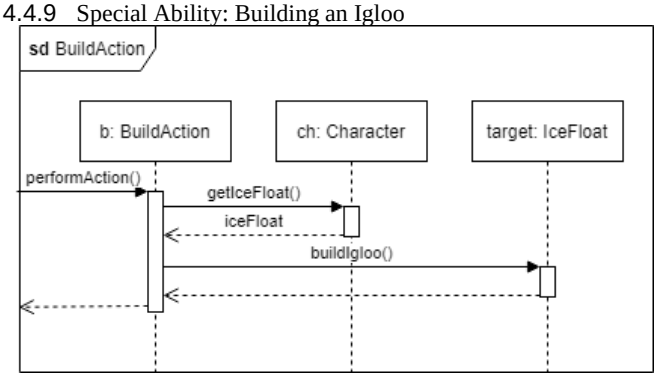
4.4.7 Move



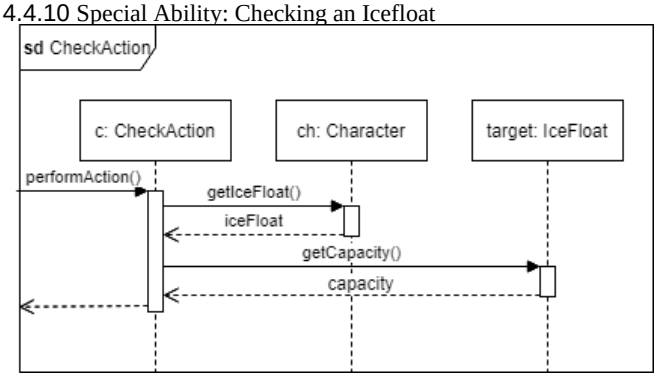
4.4.8 Fallen into Water



4.4.9 Special Ability: Building an Igloo



4.4.10 Special Ability: Checking an Icefloat



4.5 Statechart Diagramme

Die Gruppe ist der Meinung, dass keine solche Funktionalität in der Aufgabe vorkommt, wann es sich lohnen würde, sie zu modellieren.