**Operation Contract 1: clickStartGame**

**Operation:** clickStartGame()

**Cross References:** Use Case: Start Game

**Preconditions:**

* The game is launched, and the main menu is visible.

**Postconditions:**

* The game is initialised by system with initializeGame() is called.
* The game switched to Build Mode.
* The Build Mode interface, which allows the player to design dungeon halls, was displayed.

**Operation Contract 2: move**

**Operation:** move(direction: String)

**Cross References:** Use Case: Move Hero

**Preconditions:**

* The game is in Play Mode.
* The hero is placed in the hall grid.
* The move is validated by the system with validateMove().

**Postconditions:**

* If the path was clear, the position of hero was updated with updatePosition().
* If the movement was invalid (e.g., if you encounter a wall), an error message or sound was displayed.

**Operation Contract 3: placeObject”**

**Operation:** placeObject(object: String, position: String)

**Cross References:** Use Case: Build Map

**Preconditions:**

* The game is in Build Mode.
* The player has access to objects for placement.
* The object placement was validated by the system with validatePlacement().

**Postconditions:**

* If valid, the object was placed at the specified position.
* If invalid, an error message was displayed.
* The map design was saved if the player confirmed, with saveMap().

**Operation Contract 4: checkForRune**

**Operation:** checkForRune(object: String)

**Cross References:** Use Case: Check for Rune

**Preconditions:**

* The player is in a hall with objects containing a hidden rune.

**Postconditions:**

* The selected object that contained a rune was checked by system
* The door was unlocked if the rune is found by unlockDoor().
* If rune was not found, negative feedback was given by the system, allowing the player to continue searching.

**Operation Contract 5: moveToExit**

**Operation:** moveToExit()

**Cross References:** Use Case: Exit Hall

**Preconditions:**

* The player is at the exit door.
* The door is unlocked.
* The system validated the exit with validateExit().

**Postconditions:**

* The next hall was loaded with transitionNextHall().
* If it was the final hall, the game initiated the completion sequence.

**Operation Contract 6: collectEnchantment**

**Operation:** collectEnchantment(type: Integer)

**Cross References:** Use Case: Collect Enchantment

**Preconditions:**

* An enchantment is available in the game environment.
* If the enchantment is of Type 2 (storable), the player has space in their inventory.

**Postconditions (Type 1):**

* The non-storable enchantment is immediately applied via applyEnchantment(type 1).
* The player’s attributes or abilities are updated based on the effect of the enchantment.

**Postconditions (Type 2):**

* The storable enchantment is added to the player’s inventory via storeEnchantment(type 2).
* The player’s inventory is updated according to the addition of the enchantment.

**Operation Contract 7 : castEnchantment**

**Operation:** castEnchantment(type: Integer)

**Cross References:** Use Case: Cast Inventory Enchantment

**Preconditions:**

* If Type 2, the player has at least one enchantment of the specified type stored in their inventory.

**Postconditions (Type 1):**

* The non-storable enchantment is activated immediately.
* The effects of the enchantment, as an example revealing a rune or hiding from a monster, are applied for their intended duration.

**Postconditions (Type 2):**

* The storable enchantment is cast, and its effects are applied to the environment or target specified.
* If single use, the enchantment is removed from the inventory.