Table of Contents

Use Cases

Group: Fark Etmez

Use Case Name: Save Game

Use Case ID: UC-11

Scope: KUVid-Phase 2

Level: User Goal

Primary Actor: Player

Stakeholders and Interests:
-Player: Wants to save the game exactly from the paused point.
-Developer: Wants to create a game save file for the player.

Preconditions:
-Game has already been started.

Postconditions:
-The game could continue.

Main Success Scenario:
1. Player presses the save game button in the pause screen.
2. Player chooses one of the save slots which can be distinguished by “ + “ sembol
3. Game is stored completely into a database.

Extensions:
*a. At any time, File I/O error occurs.
1. Start Screen comes up with an error message.

2a. There is not enough space in the database to save the game.
1. System asks the player whether the player wants to overwrite any of the existing saved files to save the current session.

Technology and Data Variations List:
1. Database System: To save the game.

Frequency of Occurrence:
- Depends on the player’s preference. At least one occurrence is expected for a given session. However, a player can play a whole session without saving the game.

Use Case Name: Load Game

Use Case ID: UC-12

Scope: KUVid-Phase 2

Level: User Goal

Primary Actor: Player

Stakeholders and Interests:
-Player: Wants to play the game exactly from the saved point.
-Developer: Wants to provide the previously saved game to the player.

Preconditions:
-Game is saved before.
-Player logins again to play the game.

Postconditions:
-The game continued from the saved point.
-All saved features were loaded to the game.

Main Success Scenario:
1. Player presses the load game button in the start screen.
2. Player chooses one of the saved games which can be distinguished by usernames.
3. Game continues from the saved positions, scores, types/amount of atoms/molecules, positions of the objects, moving patterns and time.

Extensions:
*a. At any time, File I/O error occurs.
   1. Start Screen comes up with an error message.

2a. There are no saved games in the database.
   1. System asks the player whether the player wants to open a new game or not.

Technology and Data Variations List:
1. Database System: To load the game.

Frequency of Occurrence:
-It can occur multiple times in a game.

Use Case Name: Add Shield

Use Case ID: UC-13

Scope: KUVid-Phase 2

Level: User Goal

Primary Actor: Player

Stakeholders and Interests:
-Player: Wants to improve the efficiency of an atom by adding shields.
-Developer: Wants to add the shield to specified atom with correct spec updates.

Preconditions:
-Game is in running mode.
-An atom is placed on the shooter.

Postconditions:
-The atom has updated its specs according to the given shields.

Main Success Scenario:
1. Player chooses desired shield/s from the scoreboard with the mouse.
2. Chosen shield is applied to the atom currently placed on the shooter.
3. The atom’s specs update according to the chosen shield and its defined values.

Extensions:
*a. At any time, System fails.
1. Player can restart the game.

3a. Different kinds of shields are added to the atom.
1. Every shield has its own effects on the atom.
2. The new specs of the atom are calculated with the chaining.

Technology and Data Variations List:
1a. Player uses the mouse to pick the shield from the inventory, i.e. the stats panel.

Frequency of Occurrence:
-Could be nearly continuous. The player may select a shield and have it be applied to the selected atom at any given time.

Use Case Name: Hit a Molecule

Use Case ID: UC-2

Scope: KUVid-Phase 1

Level: User-goal

Primary Actor: Player

Stakeholders and Interests:
-Player: Wants the atom shooter to shoot an atom in an upward motion by providing an action to hit a falling molecule.
-Developer: Wants the atoms that are shot from the shooter to stay in the horizontal boundaries of the game screen.

Preconditions:
1. Game is in running mode.
2. The player has at least one atom of the randomly selected type to be shot.

Postconditions:
1. The atom moved up from the end of the atom shooter and started traveling up the game screen.
2. One atom, of whichever type was shot, was decreased from the inventory of the player.
3. The atom that was shot hit the falling molecule.

Main Success Scenario:
1. Player triggers the Shooter while an atom is selected.
2. Atom travels upwards through the game screen at a set speed.
3. Atom is traveling within the horizontal position of the compound.
4. Atom reaches the vertical position of the compound.
5. Atom collides with a molecule of the same type.
6. The atom and the molecule create a compound.
7. The newly formed compound disappears from the screen.
8. The player's score is increased by the efficiency of atom.

Extensions:
*a. At any time, System fails.
1. Player restarts the game.

2a. The atom is traveling upwards but also moving horizontally due to the angle of the shooter.
1. The atom moves vertically with the set speed but also has horizontal speed, depending on the angle of the shooter.
2. The atom then may or may not collide with the molecule to form a compound similar with the main success scenario.
   *a At any time, Atom collides with a side border of the game screen.
   1. The atom is reflected from the border.

4-5a. The atom does not collide with a molecule of the same type because there is no molecule present.
1. The atom keeps moving up the game screen

2. The atom reaches the ceiling of the game screen and disappears from the game screen and thus the player’s vision.

4-5b. The atom does not collide with a molecule of the same type but collides with a different type of molecule.

1. The atom keeps moving up the game screen.
2. The atom reaches the ceiling of the game screen and disappears from the game screen and thus the player’s vision.

Special Requirements:
Player uses a certain keyboard action to shoot atoms.
Some method to decide whether the atom and the molecule indeed collided, i.e. some sort of hit registry or collision detection.

Frequency of Occurrence: Nearly continuous, until the player runs out of atoms to shoot.

Use Case Name: Pick Atom

Use Case ID: UC-4

Scope: KUVid-Phase 1

Level: User-Goal

Primary Actor: Player

Stakeholders and Interests:
-Player: Wants to pick an atom to shoot.

Preconditions:
-Player specifies the number of each game object, the length unit and molecule structure in the building mode.
-Game is in running mode.

Postconditions:
-Desired atom was displayed on top of the shooter.

Main Success Scenario:

1. System displays a random atom on the top of the shooter.
2. Player presses the desired button.
3. System randomly selects atoms.
4. If selected atom has shields remained from the change of the atom type, System adds remaining shields to new atom.
5. System displays the selected atom on the top of the shooter.
   System repeats steps 3-5 in every step 2.

Extensions:

*a. At any time, the player is out of atoms.
  1.Player can press the "blender" icon.

*b. At any time, the game crashes.
  1.Player restarts the game.

2a. Player presses an arbitrary button.
  1.System does not change the atom.

Technology and Data Variations List:
2a. Player uses a keyboard to pick atoms.

Frequency of Occurrence: Could be nearly continuous

Supplementary Specification

Group: Fark Etmez

Introduction
This document is the repository of all KUVid 302-Phase 1 Requirements not captured in the use cases.

Functionality
The System does lots of actions in the Game time (Dropping molecules, powerups, etc.). The details are explained in Application-Specific Domain Rules.

Usability
Human Factors
The Player should be able to see all the game items easily while looking at the screen.
Colors should be distinctly different to ease understanding.
The system should give warnings when the Player does something wrong.
The system must be able to react in a short time interval to the Player's actions.

Reliability
Recoverability
If there is a failure, Player can restart the Game.

Performance
As mentioned above, our aim is to react to the Player's actions quickly. Also, the Game must not crash.

Supportability
Adaptability
The Game should adapt to changing features easily.

Configurability
The Player can configure the Game in Building Mode.

Implementation Constraints
KUVid-302 Project Team uses Java (standard Java libraries, Java Swing, etc.) to implement the Game.

Application-Specific Domain Rules

General Game Rules:

- L is the default distance unit in the game. All the dimensions are going to be driven from this unit. By default, L is 10% of the game view height. However, it is configurable in the game building mode.
- The shooter will move in L/secs.
- The width of the shooter is 0.5 L while the height is 1L.
- There are Alpha-, Beta-, Sigma-, Gamma- molecules. Each molecule has the corresponding atom, reaction blocker, and powerup.
- All have nonlinear forms. Alpha- and Beta- also have linear forms.
- There are two main behaviors while objects fall from the sky, namely:

    - Straight: falling with a speed of L/sec, perpendicular to the ground.
    - Zig-zag: falling with a speed of L/sec, but with a 45 degree angle to the vertical plane, in alternating fashion, i.e. 45 degree angle to the left followed by 45 degree angle to the right, changing direction after a distance of L is travelled.
- The molecules fall from the sky in the following manner:
    - Alpha- molecule: Zig-zag all the way through the gameview height.
    - Beta- molecule: Straight for a quarter of gameview height, zig-zag for the rest of the way.
    - Gamma- molecule: Straight for half the gameview height, zig-zag for the rest of the way.
    - Sigma- molecule: Straight throughout the gameview height.
- The reaction blockers, Alpha-b, Beta-b, Gamma-b, and Sigma-b, block the reactions between their corresponding atoms and molecules that are within 0.5L of them.
- The reaction blockers also explode when they reach the ground, creating a blast zone of 2L and destroying any object in that zone, while decreasing the player's health by a factor of (gameview width / distance to the shooter) if the shooter was in the blast zone.
- The reaction blockers follow the same pattern with their corresponding molecule while falling through the sky.
- The power-ups, also in the four types mentioned above, are used to destroy the reaction blockers.
- Power-ups are collected by the shooter being in the same place where the power-up is falling, and stored in the player's inventory. Also, the power-ups always fall in straight lines.
- To use a power-up, the player clicks on the desired power-up icon, which then appears on the atom shooter and can be shot like an atom.

<u>Building Mode Rules:</u>
- Player chooses between "easy", "medium", "hard" difficulty levels.
- Difficulty levels indicate the falling speed of objects. In easy mode the falling speed is 1 secs, in medium mode the falling speed is ½ secs, in hard mode falling speed is ¼ secs. (Falling speed indicates the object occurring speed in the screen)
- Player can specify the number of atoms, reaction blockers, powerups, molecules.
- Player can choose molecule shapes for Alpha Beta.
- Default values:
    - 100 atoms of each type
    - 100 molecules of each type and of any structure
    - 10 reaction blockers of each type
    - 20 powerups of each type

Pick Atom:
A random atom will be displayed on top of the shooter.
Whenever the player presses the "C" button on the keyboard the existing atom will change into a random atom.
The process will only work for the "C" button, there will be no change in the atom if the player presses an arbitrary button on the keyboard.

Rotate Shooter:
Shooters will be rotated if the player presses the "A" or "D" button on the keyboard.
Player presses the "A" button and the shooter will be rotated 10 degrees to the left.
Player presses the "D" button on the keyboard and the shooter will be rotated to the right.
If the shooter is already 90 degrees rotated to the left and the player presses "A", the shooter will not be rotated.

If the shooter is already 90 degrees rotated to the right and the player presses “D”, the shooter will not be rotated.
The rotation speed of the shooter is 90 degrees/sec.
This process will only work for “A” and “D” buttons on the keyboard, the shooter will not rotate if the player presses any other arbitrary button.

Powerups:
Player presses Arrow-up to shoot powerup.
The powerup number is increased by 1 when powerup is caught.
The powerup number is decreased by 1 when powerup is shot.
Powerups destroy Reaction Blockers which enter Powerup’s field (radius 0.5L) only if the type of Reaction Blocker is the same as the type of Powerup.
Powerup is placed by clicking on the Powerup icon and shot by “Arrow-Up” key.
If the Player clicks on a powerup type that Player does not have, System gives a warning.
Powerup must be placed at the top of Shooter after picking.

Hit a Molecule:
The shooter will be triggered with the Up-Arrow.

Move Shooter:
Shooter will be moved left or right as the player presses the “Left Arrow” or “Right Arrow” buttons on the keyboard.
Player presses the “Left Arrow” key and the shooter will be moved to the left as long as the key is pressed.
Player presses the “Right Arrow” key and the shooter will be moved to the right as long as the key is pressed at the speed of L/sec.
If the shooter is at most left of the game screen and the player presses the “Left Arrow”, the shooter will not be moved.
If the shooter is at most right of the game screen and the player presses the “Right Arrow”, the shooter will not be moved.
The moving speed of the shooter is L/sec.
This process will only work for “Left Arrow” and “Right Arrow” buttons on the keyboard, the shooter will not move if the player presses any other arbitrary button.

Efficiency:

Atom-Molecule-Compound Values:

Alpha (8 protons. 7,8,9 neutrons)
Alpha stability constant = 0.85
Efficiency = (1 - (| # of neutrons - # of protons | / # of protons)) * Alpha stability constant

Beta (16 protons. 15, 16, 17, 18, 21 neutrons.)
Beta stability constant = 0.9
Efficiency = Beta stability constant - (0.5 * | # of neutrons - # of protons | / # of protons)

Gamma (32 protons. 29, 32, 33 neutrons.)
Gamma stability constant = 0.8
Efficiency = Gamma stability constant + (| # of neutrons - # of protons | / (2 * # of protons))

Sigma (64 protons. 63, 64, 67 neutrons.)
Sigma stability constant = 0.7
Efficiency = (1 + Sigma stability constant) / 2 + (| # of neutrons - # of protons | / # of protons)

Shield Values:

Eta:
if # shielded atom neutrones != # shielded atom protones:
Efficiency+= (1 - shielded atom efficiency) * |# shielded atom neutrones - # shielded atom protones| / # shielded atom protones.
Otherwise:
(1 - shielded atom efficiency) * Eta_efficiency_boost
Eta_efficiency_boost = 0.05

Lota:
Efficiency+= (1 - shielded atom efficiency) * Lota_efficiency_boost
Lota_efficiency_boost = 0.1

Theta:
Efficiency+=(1 - shielded atom efficiency) * Theta_efficiency_boost
Theta_efficiency_boost = is random between 0.05 and 0.15)

Zeta:

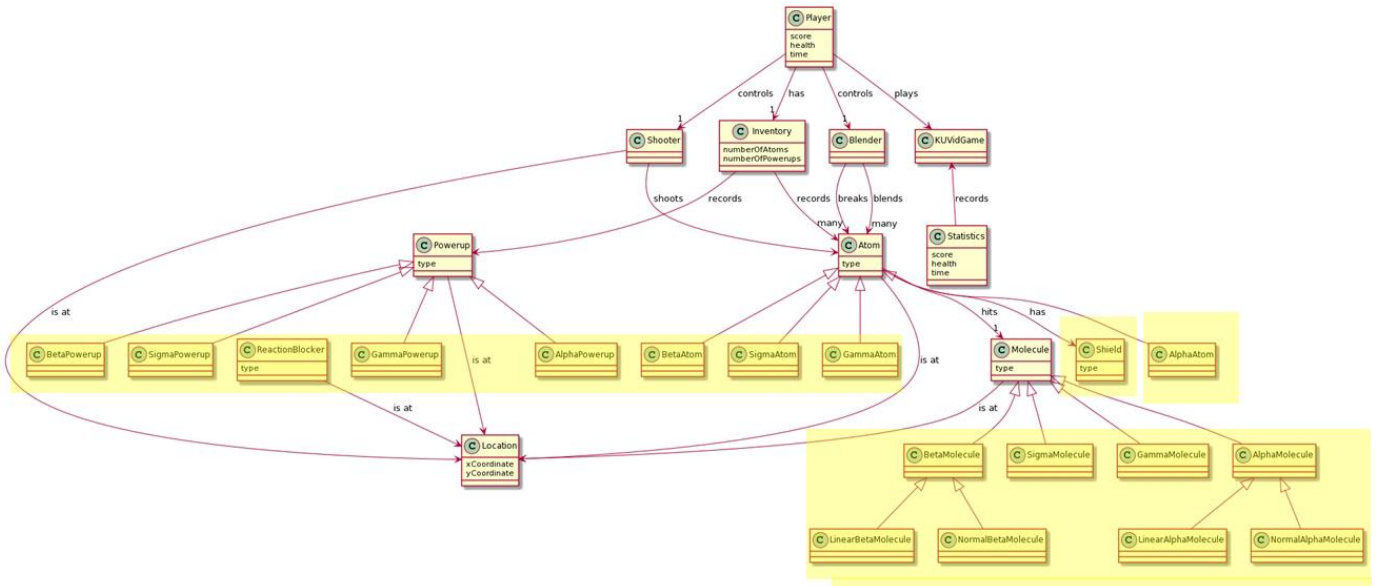
Efficiency+=(1 - shielded atom efficiency) * Zeta_efficiency_boost
(Zeta_efficiency_boost = 0.2)
Zeta improves the efficiency iff
# shielded atom protons = # shielded atom neutrons
The score increment when a shielded atom hits a corresponding molecule, as for normal one, is equal to its efficiency

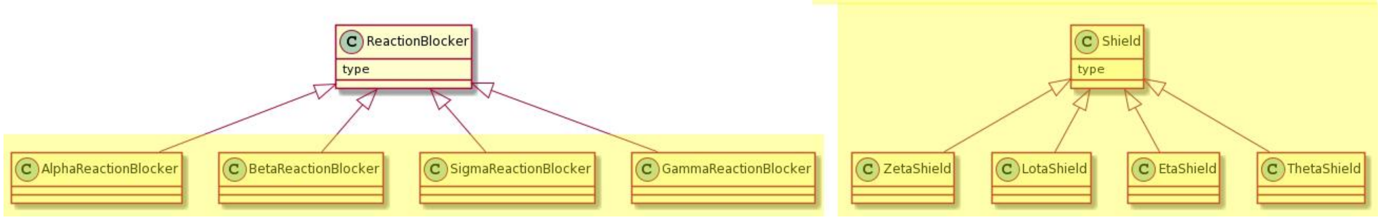
The shields affect the speed of the shielded atom as well.
Each Eta shield reduces the speed by 5%.
Each Lota shield reduces the speed by 7%.
Each Theta shield reduces the speed by 9%.
Each Zeta shield reduces the speed by 11%

In the case of combined shields, the speed is calculated by chaining the effects of the shields.

Domain Model

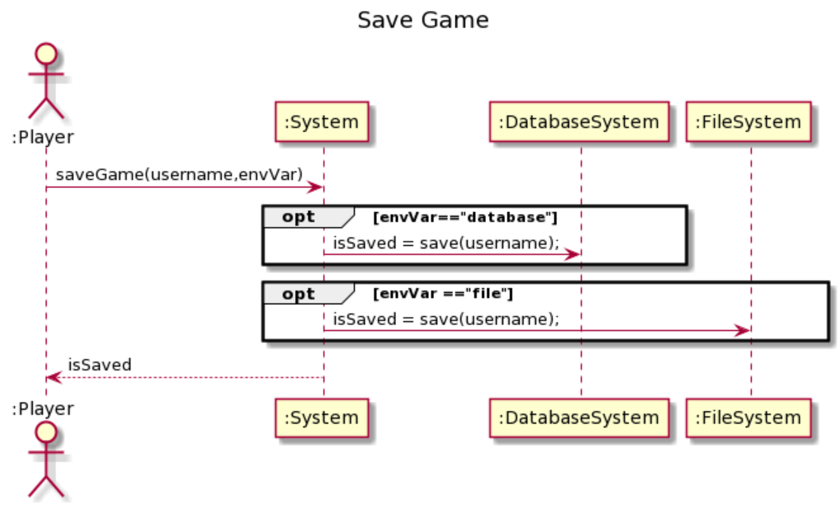
Group: Fark Etmez

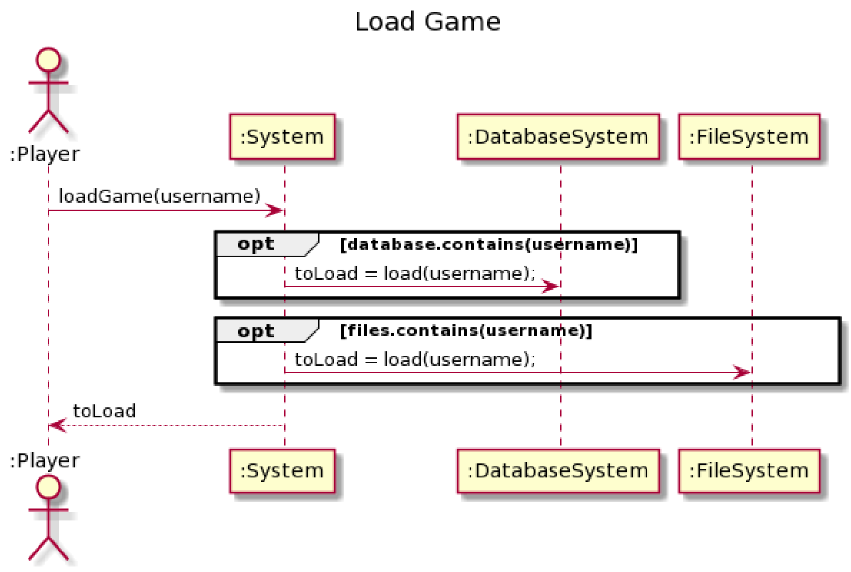


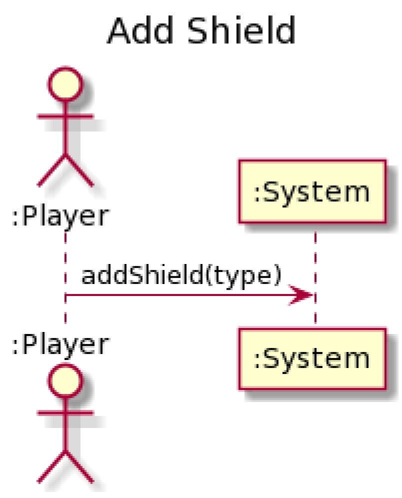


System Sequence Diagrams

Group: Fark Etmez







Operation Contract

Group: Fark Etmez

Operation:	saveGame(username, envVar)
Cross References:	***Use Cases:*** Save Game
Preconditions:	Game has already been started
Postconditions:	-a GameSaveObject (gso) was created
	-gso was associated with the username
	-gso was associated with the timeLeft
	-gso was associated with the score
	-gso was associated with gameScreen objects.
	-gso was saved to the desired storage service according to envVar as a completed save.