5 Girls 1 Boi
Final Report
COMP 302 Term Project
KUvid Game

Table Of Contents

Introduction

Throughout the Fall 2020 term, we have developed the KUvid Game as the term project of Comp 302. We have implemented this project by following the Model-View Separation, GRASP, and GOF Patterns that we have acquired during the classes. These patterns and Model-View Separation Principle made the game easier-to-update and made it possible to use on many devices as the domain layer is not UI-dependent.

We have used git to code and work on different tasks simultaneously. Furthermore, we have communicated through zoom both with our TA and as a group.

In the end, we tested some features of the game by using JUnit and tested the whole game by playing ourselves as well.

Vision

Version	Date	Description	Authors
Initial Draft	11.November.2020	First draft. To be refined primarily during elaboration	Elif Satır Irmak Tahtakılıç
Phase 1 Demo	10 December 2020	Draft of Phase 1. To be refined primarily during elaboration	Elif Satır Irmak Tahtakılıç
Phase 2 (Final Version)	17 January 2021	Final version of the Project	Elif Satır Dilara Deveci

Introduction

We envision a fun and interactive game called, KUvid Game, with the flexibility of saving and loading the game from a file or a database.

Positioning

Problem Statement

Implementing KUvid game in accordance with Model-View Separation, GRASP, and GOF Patterns.

Product Position Statement

The game is for users who enjoy playing PC games. With its outstanding and original story, KuVid is going to take its place next to high-level games in the industry as a game that can be played on various platforms.

Terse summary of who the system is for, its outstanding features, and what differentiates it from the competition.

Alternatives and Competition

There are many alternatives and competition to KUvid Game as there are many similar PC games in the market.

Stakeholder Description

Keeping in mind the age range of the players and their preferences.

Key High-Level Goals and Problems of the Stakeholders

A requirements workshop with the development team and other stakeholders led to the identification of the following key goals and problems:

Consolidate input from the Actor and Goals List, and the Stakeholder Interests section of the use cases.

User-Level Goals

The users (and external systems) need a system to fulfill these goals:

Player: player/user, who controls the game. Its ultimate goal is to get the highest score s/he can get in the given time and without running out of the health points as fast as possible.

KUvid Game: the 2D gaming environment the user can see.

User Environment: KUvid game itself, the game frame

External Systems:

MongoDB: for saving and loading to/from database

Local Files: for saving and loading to/from the local file system as a txt file

Product Overview

Summary of System (KUvid Game) Features

- Build the game
- Move the shooter
- Rotate the shooter
- Collect a molecule
- Collect a powerup
- Destroy a reaction blocker
- Blend atoms
- Break atoms

- Pause the game
- Resume the game
- Save the game
- Load the game
- Pick an atom
- Shoot an atom
- Pick a powerup
- Specify number of items (atoms, blocker, powerups, molecules, length unit, structure)
- Choose molecule animation
- Specify difficulty
- Shield an atom

Teamwork Organization

- Use Case Diagrams: Atakan Küpeli, Yağmur Yıldırım
- Use Case Narratives: Atakan Küpeli, Yağmur Yıldırım
- Domain Model: Irmak Tahtakılıç, Elif Satır
- System Sequence Diagrams: Fulya Akın
- Operation Contracts: Dilara Deveci
- Glossary, Vision and Supplementary Specifications: Irmak Tahtakılıç, Elif Satır

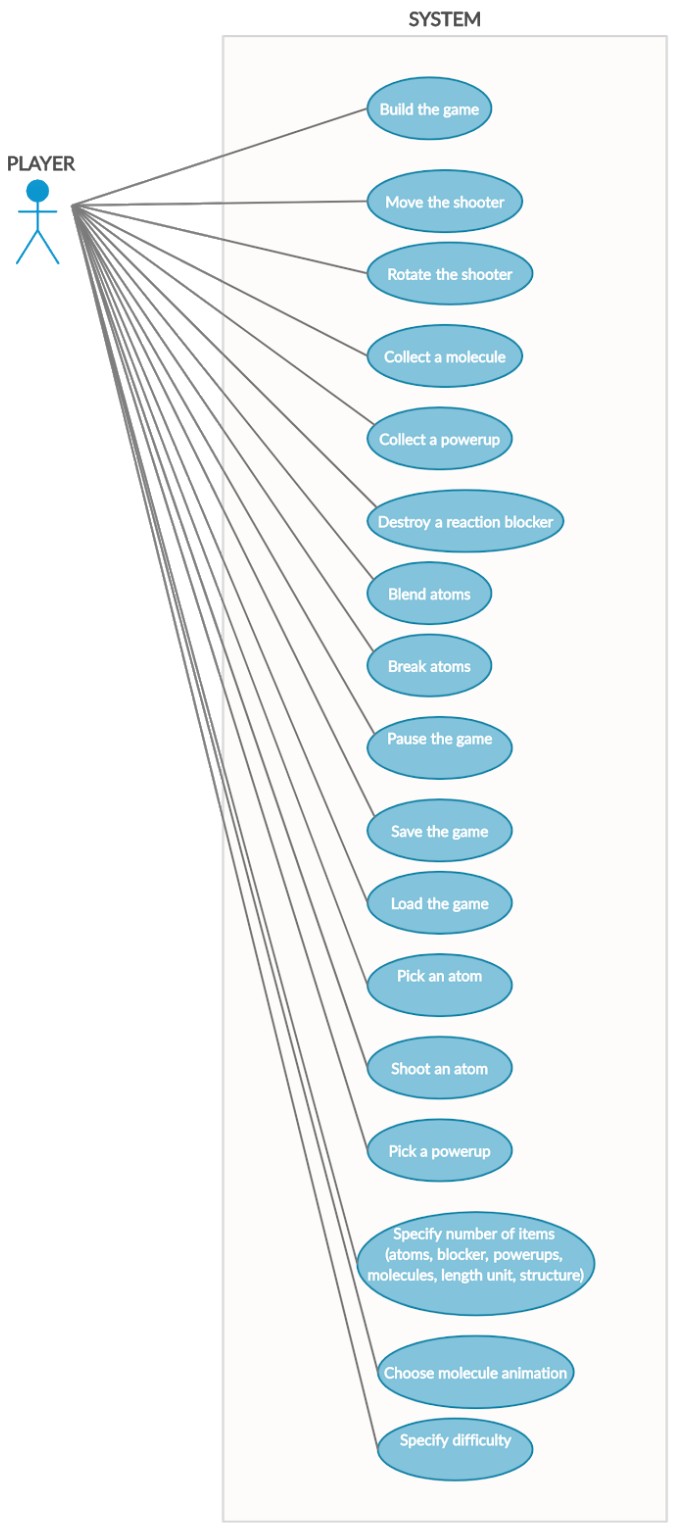
- Sequence Diagrams: Dilara Deveci, Elif Satır
- Communication Diagrams: Dilara Deveci, Elif Satır
- Class Diagram: Atakan Küpeli, Yağmur Yıldırım
- Logical architecture (UML Package diagrams): Fulya Akın
- Discussion of the Design Alternatives: Irmak Tahtakılıç
- Building Frame (the required code for Design I): Fulya Akın

- Observer Pattern: Dilara Deveci, Atakan Küpeli
- Factory Patterns: Dilara Deveci
- Controller Pattern: All
- Singleton Pattern: Atakan Küpeli
- Adapter: Dilara Deveci
- Decorator: Yağmur Yıldırım, Fulya Akın

- Shield Implementation: Fulya Akın, Yağmur Yıldırım
- Animation: Irmak Tahtakılıç, Elif Satır

Overall, we all worked in coherence and contributed to each other's work load to keep in track of where we are in the game and to gain a faster understanding. We helped each other to develop the game and exchanged the codes through git.

Use Case Diagrams



Use Case Narratives

Use Case UC1: Build the game

Scope
Kuvid Game

Level
User goal

Primary Actor
Player

Stakeholders and Interests
Player: Wants to specify the game settings and play the game.

Preconditions
The program successfully launched and the input prompts are shown.

Success Guarantee/Postconditions
The game is successfully launched with the given parameters.

Main Success Scenario

1. The player starts the program.
2. Building mode launches.
3. The player specifies the number of atoms.
4. The player specifies the number of reaction blockers.
5. The player specifies the number of powerups.
6. The player specifies the number of molecules.
7. The player specifies the number of shields.
8. Player chooses the spinning animation of the molecules.
9. The player selects the difficulty of the game.
10. Building mode ends and the game starts.

Extensions

* a. At any time, the game crashes.
1. Player restarts the game, building mode launches.

3-7a. Player enters invalid number.
1. Game sets the default value if the field is left empty..

Special Requirements

- Computer screen, keyboard, and touchpad/mouse hardware is required.

Technology and Data Variations List

1a. Program launched by keyboard or touchpad.
3-7a. Number of items entered by keyboard.
8-9a. Specification chosen by keyboard or touchpad/mouse.

Frequency of Occurrence
At the beginning of the game.

Open Issues
- Are there any upper and lower limits to the parameters to be considered invalid?

Use Case UC4: Collect a molecule

Scope
Kuvid Game

Level
User goal

Primary Actor
Player

Stakeholders and Interests
Player: Wants to hit a molecule using an atom and increase their score.

Preconditions
Player has an atom to shoot in their inventory.
There is at least one molecule in the screen.
Player has enough health points to continue the game.
Player didn't reach the time limit.

Success Guarantee/Postconditions
The atom is shot and collides with the targeted corresponding molecule.
The number of atoms in the player's inventory has decreased.
The score of the player increased.
The molecule disappears from the screen.

Main Success Scenario
1. The player starts the game.
2. In building mode, the player specifies the initial settings of the game.
3. The player selects the atom to be shot.
4. The game places the selected atom on the shooter.
5. The player moves the shooter either horizontally or angularly.
6. The player shoots the atom.
7. Game checks for the collision of the atom with the corresponding molecule.
8. The atom and the molecule disappear when they collide.
9. Player's number of specified atoms decreases and their score increases.

Extensions
*a. At any time, the game fails:

1. Player restarts the game.

2a. Player enters invalid number.

2. Game asks for a valid entry.

3-9a. Player's shooter collides with a reaction blocker.

1. Player's health point is reduced.
- If the player's health becomes 0, the player dies, the game ends and the score is displayed.

6b. Player fails to collide the atom with the molecule.

1. Score does not increase.
2. Player's number of atoms of that type decreased.
- If the player's atom number becomes 0, the game ends and the score is displayed.
3. The molecule does not disappear from the screen.

7a. The atom and the molecule are not of the same type.

1. Score does not increase.
2. Player's number of atoms of that type decreased.
- If the player's atom number becomes 0, the game ends and the score is displayed.
3. The molecule does not disappear from the screen.

Special Requirements

- Computer screen, keyboard, and touchpad/mouse hardware is required.

Technology and Data Variations List

1a. Program launched by keyboard or touchpad.
2a. Specification chosen by keyboard or touchpad/mouse.
3-6a. Button pressed on keyboard.

Frequency of Occurrence

During the game.

Open Issues

Use Case UC7: Blend atoms

Scope
Kuvid Game

Level
User goal

Primary Actor
Player

Stakeholders and Interests
Player: Wants to increase the number of wanted atoms by using other specified atoms.

Preconditions
Player has needed atoms to blend in their inventory.
Player has enough health points to continue the game.
Player didn't reach the time limit.

Main Success Scenario

1. The player starts the game.
2. In building mode, the player specifies the initial settings of the game.
3. The player selects a type of atom to blend another type of atom.
4. The player presses the blender to blend the atoms.
5. The number of atoms blended increases, the number of atoms broken decreases.

Extensions

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

2a. Player enters invalid number.
   1. Game asks for a valid entry.

3-4a. Player's shooter collides with a reaction blocker.
   1. Player's health point is reduced.
   - If the player's health becomes 0, the player dies, the game ends and the score is displayed.

3b. Player does not have the type of atom they want to blend.
   1. Blender does not work.

Special Requirements

- Computer screen, keyboard, and touchpad/mouse hardware is required.

Technology and Data Variations List

1a. Program launched by keyboard or touchpad.
2a. Specification chosen by keyboard or touchpad/mouse.
3-4a. Button pressed on keyboard.

Frequency of Occurrence

During the game.

Open Issues

Use Case UC11: Save the game

Scope
Kuvid Game

Level
Subfunction

Primary Actor
Player

Stakeholders and Interests
Player: Wants to save the game.

Preconditions

Player has enough health points to continue the game.
Player didn't reach the time limit.

Success Guarantee/Postconditions
The game is saved to either a local file system or a database system specified in a Java environment variable.

Main Success Scenario

1. The player starts the game.
2. In building mode, the player specifies the initial settings of the game.
3. The player plays the game without running out of health or time.
4. Player saves the game.
5. All game information including player username, types of atoms and molecules, position (e.g., coordinate of the molecules), score, movement patterns, remaining time is sent to the database.

Extensions

*a. At any time, the game fails:
  1.Player restarts the game.
2a. Player enters invalid number.
  1.Game sets default value if field is left empty..
3a. Player's shooter collides with a reaction blocker.
  1.Player's health point is reduced.
  - If the player's health becomes 0, the player dies, the game ends and the score is displayed.

4a. Game fails to save.
  1.Error message shows on the screen.
5a. Game is saved to the local file system.
5b. Game is saved to the database through API.

Special Requirements

- Computer screen, keyboard, and touchpad/mouse hardware is required.

Technology and Data Variations List

1a. Program launched by keyboard or touchpad.
2a. Specification chosen by keyboard or touchpad/mouse.
3-7. Button pressed on keyboard.

Frequency of Occurrence
When a player wishes to save.
Open Issues

Use Case UC13: Shield an atom

Scope
Kuvid Game

Level
User goal

Primary Actor
Player

Stakeholders and Interests
Player: Wants to shield an atom.

Preconditions
Player has enough health points to continue the game.
Player didn't reach the time limit.
Player has enough shields and atoms.

Success Guarantee/Postconditions
Player's selected atom is shielded.

Main Success Scenario

1. The player starts the game.
2. In building mode, the player specifies the initial settings of the game.
3. The player plays the game without running out of health or time.
4. Player decides on the shield they wish to use.
5. Player pressed appropriate buttons to shield their chosen atom.

Extensions

*a. At any time, the game fails:
 1.Player restarts the game.
2a. Player enters invalid number.
 1.Game sets default value if field is left empty..
3a. Player's shooter collides with a reaction blocker.
 1.Player's health point is reduced.
 - If the player's health becomes 0, the player dies, the game ends and the score is displayed.
5a. Player stacks shields on the desired atom.
5b. Player changes atoms.
 1.The shield that is placed remains on the previous atom.

Special Requirements

- Computer screen, keyboard, and touchpad/mouse hardware is required.

Technology and Data Variations List

1a. Program launched by keyboard or touchpad.
2a. Specification chosen by keyboard or touchpad/mouse.
3-5a. Button pressed on keyboard.

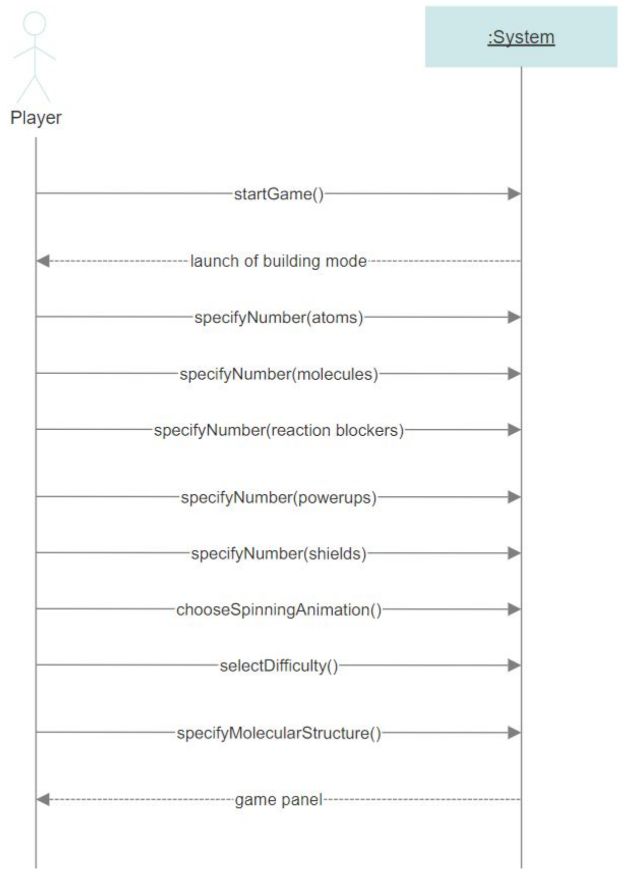
Frequency of Occurrence

When a player wishes to save.

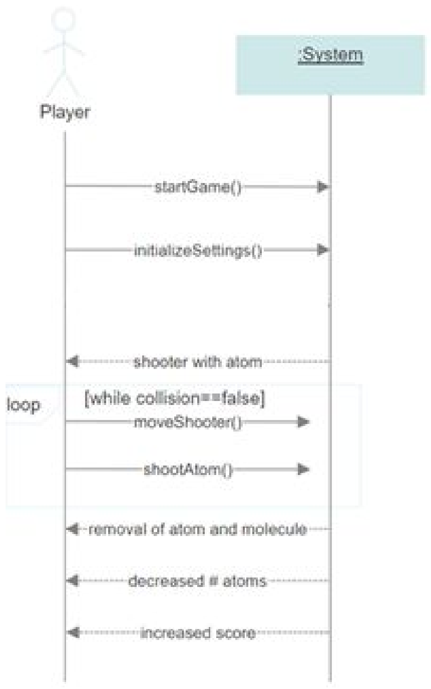
Open Issues

System Sequence Diagrams

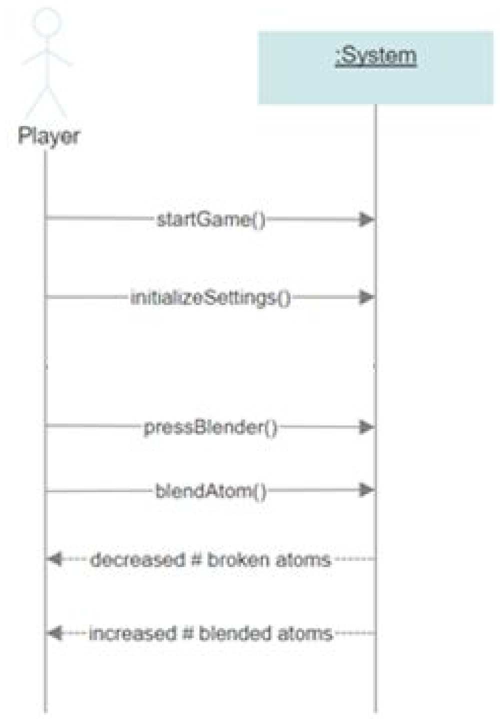
Use Case UC1: Build the game



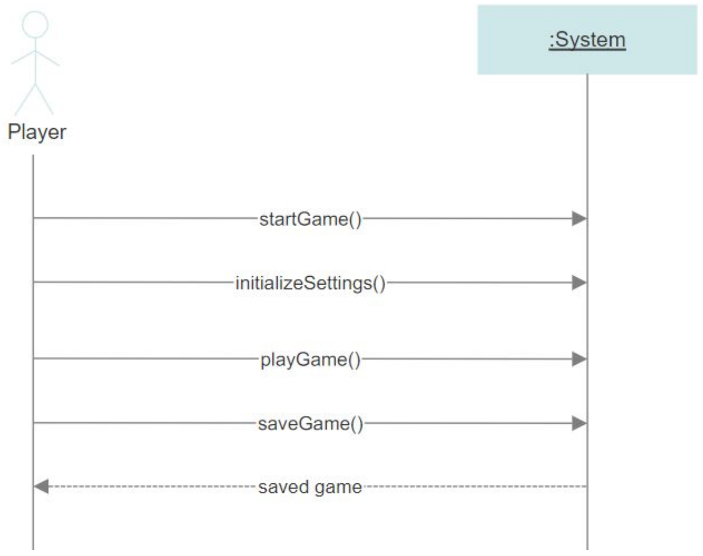
Use Case UC4: Collect a molecule



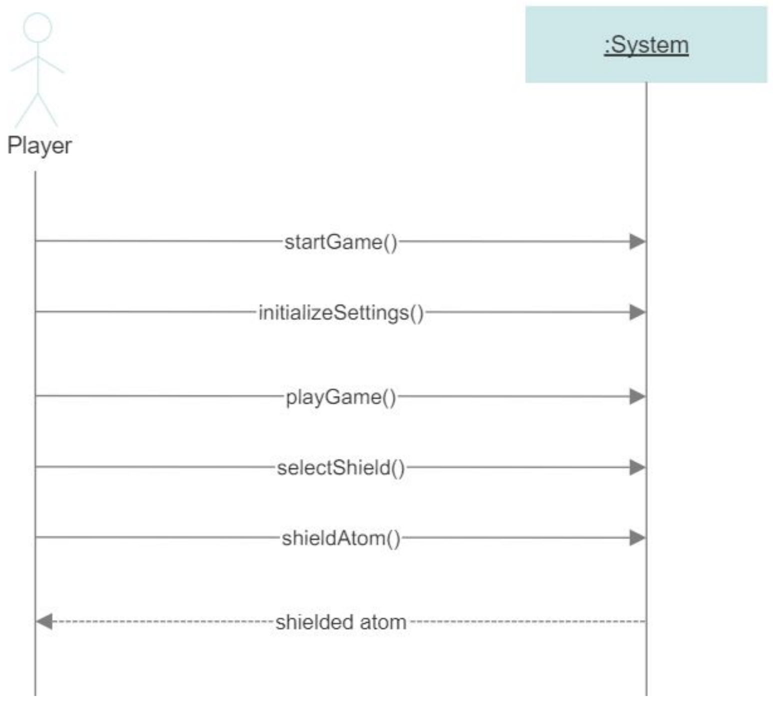
Use Case UC7: Blend atoms



Use Case UC11: Save the game



Use Case UC13: Shield an atom



Operation Contracts

OC 1

Operation: initializeSettings(atoms: Int, molecules: Int, blockers: Int, powerups: Int, shields: Int, difficulty: String, molecularStructure: String)
Cross References: Build the game
Preconditions: player starts the game
Postconditions:

-An instance of Kuvid Game, ku was created.
-ku.AtomNo became atoms.
-ku. MoleculeNo became molecules.
-ku.BlockerNo became blockers.
-ku.PowerupNo became powerups.
-ku.ShieldNo became shields.
-ku.difficulty became difficulty.
-ku.molecular structure became molecularStructure
-ku was associated with the player

OC 4

Operation: shotAtom(atom: Atom)

Cross References: Collect a molecule
Preconditions: There is enough health points, time and at least one atom to shoot
Postconditions:

-An atom instance, atm was created.
-atm was associated with the shooter.
-atm was associated with a molecule it collided with.
-atm and that instance of molecule was deleted.

OC 7

Operation: blendAtom()
Cross References: Blend atoms
Preconditions: there are enough atoms to blend and create the desired type of atom
Postconditions:

-the instances of atom, atoms, was associated with the Blender.
-An instance of atm was created.
-atm.type became type of created atom.
-atm was associated with the shooter.
-the instances of atoms that had been blended, were deleted

OC 11

Operation: saveGame()
Cross References: Save the game
Preconditions: There is an instance of KuvidGame that is being played by the Player, p
Postconditions:

-A new local file, fl was created
-p was associated with fl

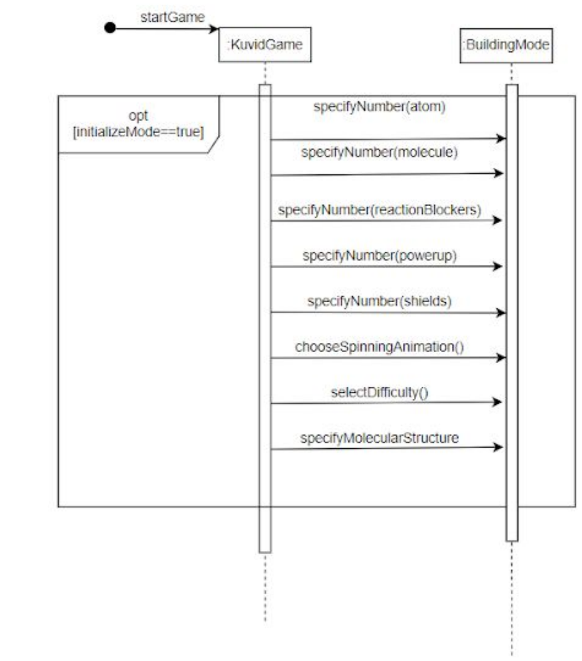
OC 13

Operation: shieldAtom()
Cross References: Shield an atom
Preconditions: There is at least one shield and an instance of atom, atm that is associated with the shooter.
Postconditions:

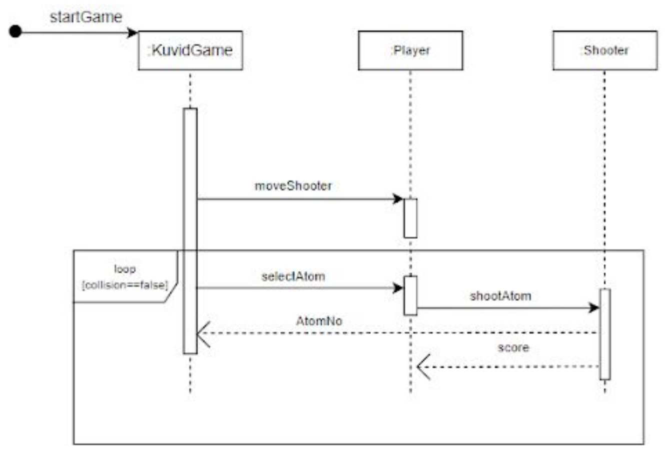
-An instance of shield, sld was created
-The atom, atm was associated with sld

Sequence Diagrams

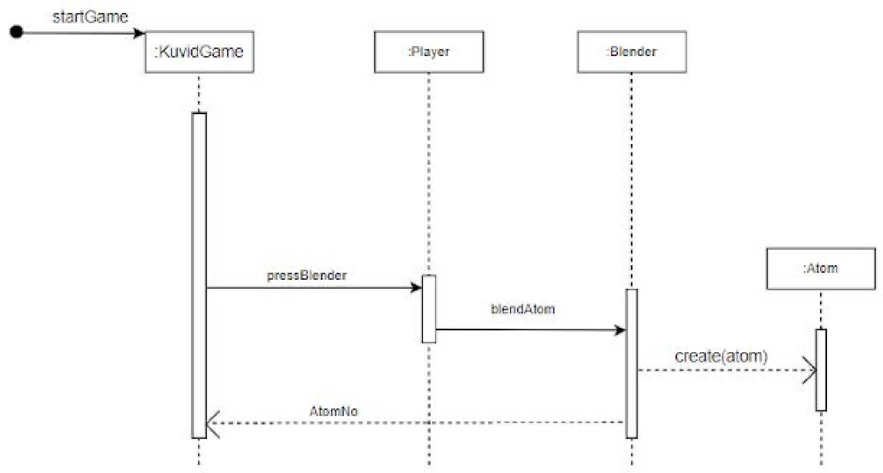
UC1: Build the game



UC4: Collect a molecule

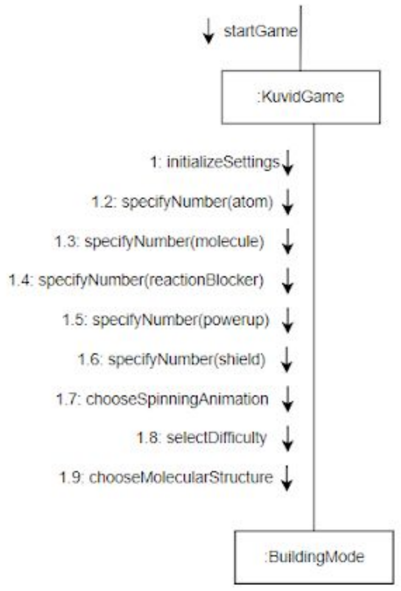


UC7: Blend atoms

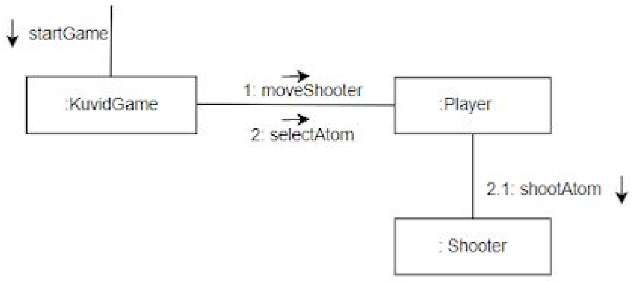


Communication Diagrams

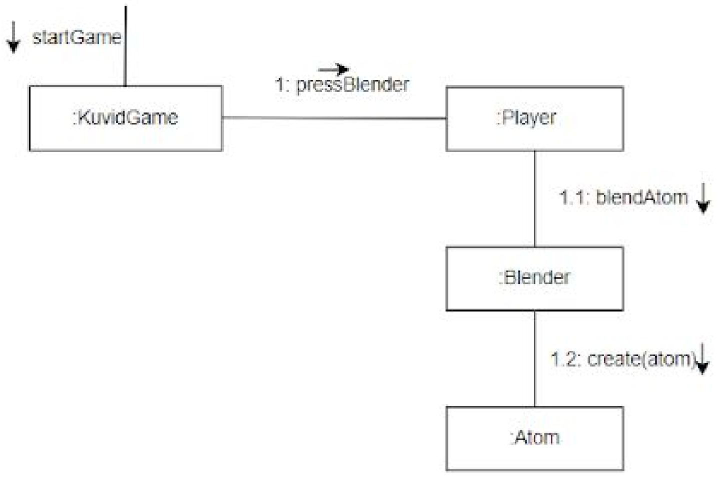
UC1: Build the game


startGame
:KuvidGame
1: initializeSettings
1.2: specifyNumber(atom)
1.3: specifyNumber(molecule)
1.4: specifyNumber(reactionBlocker)
1.5: specifyNumber(powerup)
1.6: specifyNumber(shield)
1.7: chooseSpinningAnimation
1.8: selectDifficulty
1.9: chooseMolecularStructure
:BuildingMode

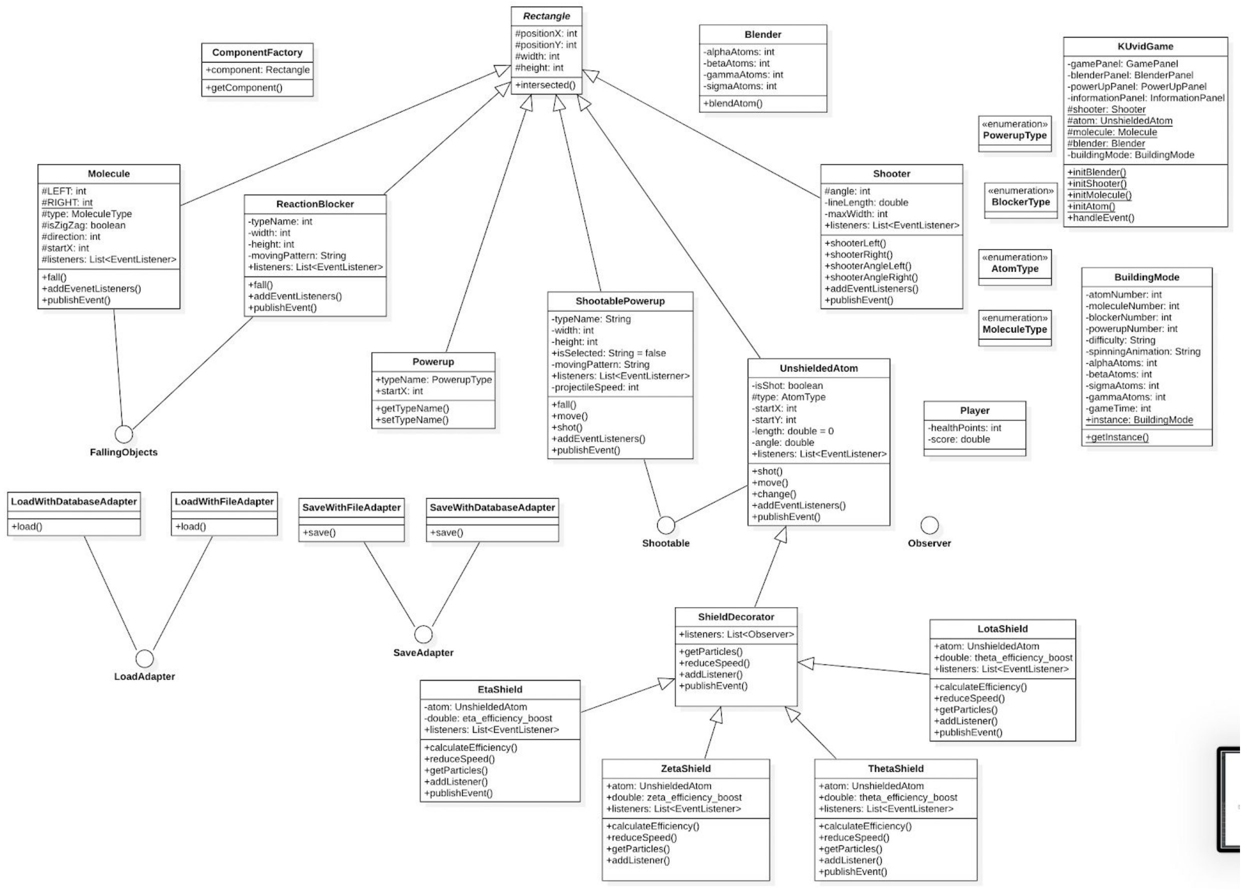
UC4: Collect a molecule



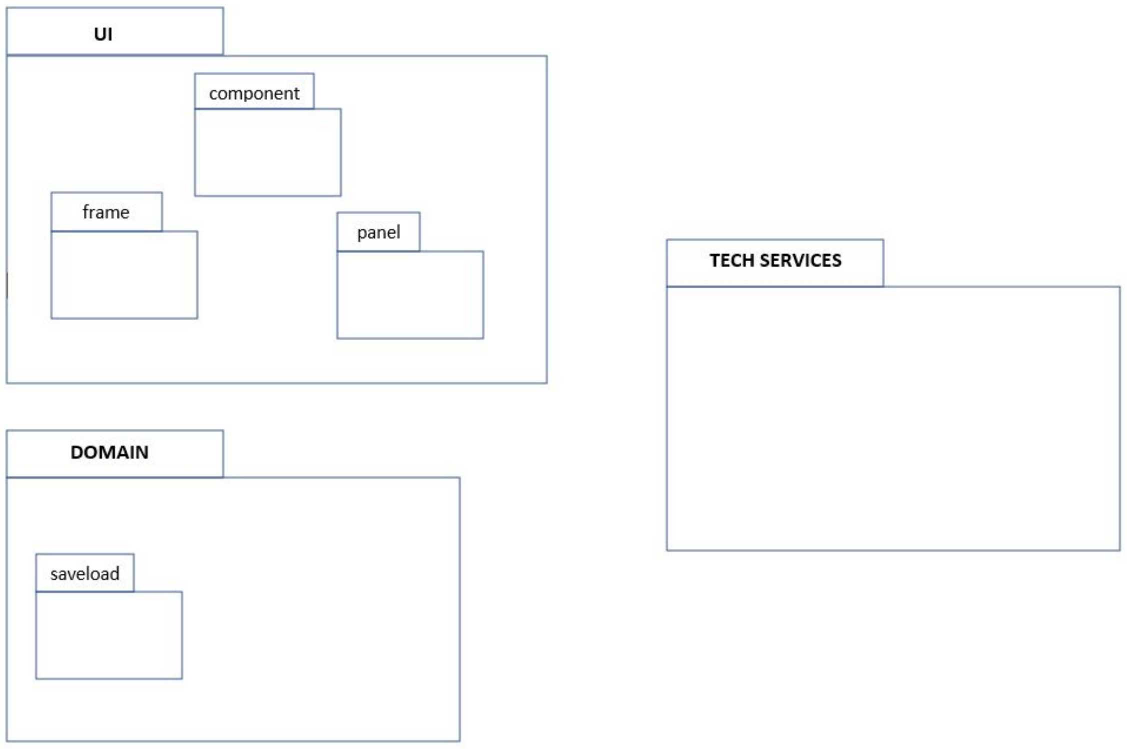
UC7: Blend atoms



Class Diagram



Package Diagram



Discussion of Design Alternatives and Design Patterns and Principles

GRASP Principles

Low Coupling Principle

It is a measure of how strongly one element is connected to, has knowledge of, or depends on other elements. It answers “How to reduce the impact of change on software?”.

PRO’S

- It reduces time, effort and defects involved in modifying software
- Easier to to understand in isolation
- Easier to re-use
- It does not create problems in areas where change is likely

Application
Shooting a Powerup: In order to see if the collusion really occurred we needed to get the information from the Player but if we added this information to the player, we would not have low coupling so we added this responsibility to the Shooter so we had the low coupling principle. As an alternative we could assign these to KUVidgame class but since it was the expert and it was undesirable. The solution suggested by Expert is undesirable, usually because of problems in coupling and cohesion.

High Cohesion Principle

It answers the question “How to keep objects focused, understandable, and manageable, and as a side effect, support low coupling?”.

PRO’S

- Easier to comprehend
- Easier to reuse
- Easier to maintain
- It is not constantly affected by change

Application
Building the game: Initialize settings in Buildingmode named class and added into its responsibility. These responsibilities could be given to KUVid game, resulting in too many self-references. In order to avoid this we created a separate class which is BuildingMode Class.

Controller Pattern

It is a delegation pattern which answers the question “What first object after or beyond the UI layer should receive the message from the UI layer?”

PRO’S

- Control class is only responsible from system-level operations
- It helps identify out-of-sequence operations

- It increased potential for reuse and pluggable interfaces
- Opportunity to reason about the state of the use case

CON’S
- Over assignment of responsibility
- Controller may suffer from low cohesion

Application
We made KUvidGame and BuildingMode classes our controllers which are the only domain classes communicating with the UI layer.

GoF Patterns

Singleton Pattern

An implication of the pattern is that there is only one instance of a class instantiated never two. Objects need a global and single point of access.

PRO’S
- A developer has global visibility to this single instance

CON’S
- Creation work (possibly holding on to expensive resources) avoided if instance never created.
- Static methods are not polymorphic, don’t permit overriding.
- Object-oriented remote communication mechanisms only work with instance methods
- Static methods are not remote-enabled.

Application
There will be one instance which is used in the BuildingMode class.

Factory Pattern

It answers the question “Who should be responsible for creating objects when there are special considerations, such as complex creation logic, a desire to separate the creation responsibilities for better cohesion, and so forth?”.
Factory methods return objects typed to an interface rather than a class, so that the factory can return any implementation of the interface, also accessed with Singleton Pattern.

PRO’s
- It separates the responsibility of complex creation into cohesive helper objects.
- It hides potentially complex creation logic.
- It allows introduction of performance-enhancing memory management strategies, such as object caching or Recycling.

Application
FallingObjectsFactory class and also its FallingObjects interface. We have chosen this to maintain a separation of concerns; in which there are two different types for objects to fall “spinning” or “straight”. This was also useful for applying the High Cohesion Principle.

Observer

It generally provides a way to loosely couple objects in terms of communication.

PRO'S

- It supports low coupling
- It supports Model-View Separation since it is used for connecting a non-UI object to a UI object

Application
Created in Domain with Observer interface called Observer which has method update so it would check the updates and notify the other classes in case of having an update. Also used EventListeners to see if there was an update.

Adapter

Adapters use interfaces and polymorphism to add a level of indirection to varying APIs in other components.

PRO'S

- Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.
- Provides a stable interface to similar components with different interfaces

Application
A package called saveload is added in the domain which uses adapter which is for saving and loading the game.

Decorator

In this approach, the concern (such as security) is decorated onto other objects with a Decorator object that wraps the inner object and interposes the service.

PRO'S

- It provides a flexible alternative to subclassing for extending functionality
- It allows behavior modification at runtime rather than going back into existing code and making changes.

Application
Used for creating shields in class name ShieldDecorator.

Supplementary Specifications

Introduction

This document is the repository of all KUvid requirements not captured in the use cases.

Functionality

Logging and Error Handling

Log all errors to persistent storage.

Player Action Rules

Shooting the atom/ powerup: Arrow up.
Moving the shooter: left, right arrows.
Rotating the shooter: “A” rotates to left by 10 degrees, “D” rotates right by 10 degrees.
Picking an atom to shoot: “C” button switches the atom on the top of the shooter randomly.
Pausing, resuming the game: “P”, “R”.
Saving or loading a game: “S”, “L” (the game has to be paused to do these actions).
Blending/Breaking atoms: to blend or break an atom from a certain rank to another(atoms ranks are: Alpha:1, Beta:2, Gamma:3, Sigma:4), first press “B”, then the type the rank of the source
atom then the rank of the atom to be produced.
Picking a powerup: Clicking its icon on the screen, it will appear at the top of the shooter like the atom. Then it can be shot using Arrow-up. To be used, they need to be first caught and saved by the player. To collect a powerup, the shooter needs to shoot an atom and the atoms need to collide with it just like the molecules.
Choosing a shield: “L” for Lota, “Z” for zeta, “T” for theta and “E” for eta.

Score Calculation Formula

Previous score + (1 / gameTimePassed) + efficiency of the atom

Shooter

The gun can be rotated 180 degrees. The width of the vehicle is 0.5 L and the height is 1 L. The horizontal movement speed of the shooter is L/sec. The rotation speed of the gun is 90 degrees/sec. The shooter is located at the bottom of the screen. The atom in the shooter’s gun is selected randomly from the available atoms, it can be changed as explained in the “Player Actions” section. It also has 100 Health Points at the beginning of the game.

Molecules

They are named by the atoms they require to stay stable in the earth environment, namely: Alpha-, Beta-, Gamma- and Sigma-. They initially appear at the top of the screen and fall towards the bottom. Due to having different weights, and due to the variable resistance of the air at different heights; these molecules will adopt different movement patterns while falling. We have two movement patterns:

- Straight: falling with a speed of L/sec with an angle of 90 degrees.
- Zig-Zag: Falling with the speed of L/sec as well but with 45 degrees alternating fashion. First 45 towards right then towards left, then right, then left …etc. Before altering the direction a distance of L is to be traveled.

Since Alpha- are the lightest molecules, they will follow the zig-zag pattern until reaching the ground. Beta- will fall straight until passing one-quarter of the game view height then start zig-zag. Gamma- falls straight until passing the half of the game view height then starts zig-zag. While Sigma- falls straight from the beginning till reaching the ground.

The width of a molecule is 25% of L and the height is 25% of L. For Alpha- and Beta-, the molecule-structures are customizable as will be explained in the "building mode".
To collect a molecule (form a compound) the atom shot by the shooter should hit the bounding box of that molecule.

Atoms

There are four types of atoms: Alpha, Beta, Gamma, and Sigma. Each of these atoms can be shot by the shooter. The atoms travel at the speed of L/sec, in straight lines, and with an initial angle depending on the shooter's gun direction. When colliding with the borders of the game view they are reflected with an angle of 90 degrees. The diameter of the atom is 10% of L.

Reaction Blockers

They fall from the top of the screen. They can prevent the reaction to form the needed stable materials. They can as well harm the player if they fall on the ground. They have a surrounding field, represented by a circle centered, the radius of the circle is described as destruction radius, at the blocker and having the radius of 0.5L. These blockers are named by their type as Alpha-b, Beta-b, Gamma-b, Sigma-b. Each blocker blocks the unification of the corresponding atom and molecule and destroys them if they enter its field. If the blocker falls on the ground it explodes (or touches the shooter), the explosion field is circular with the destruction radius increased to 2L. Any molecule or atom in this scope will be destroyed. And the player's health can be reduced by a factor of (game view width/distance between the shooter and the blocker) if the shooter was in the field of the explosion. Reaction blockers have the same movement patterns as the corresponding molecules. The dimensions of them are also the same.

Powerups

Each of them can help in destroying certain blockers. They are +Alpha-b, +Beta-b, +Gamma-b, +Sigma-b. To be used, they need to be first caught and saved by the player. To collect a powerup the atom shot by the shooter should hit the bounding box of that molecule. Powerups fall in straight lines always. The powerups can be shot the same way as atoms. The power-up can destroy the corresponding blocker if it enters its field.

Blender

The blender can be used to:

- Blend atoms: two Alpha atoms to get Beta, 3 to get Gamma, 4 to get Sigma. Or blend 2 Betas to get Gamma, 3 to get Sigma. Or blend 2 Gamma to get Sigma.
- Break atoms: the number of atoms produced by breaking an atom can be inferred by reversing the blending rules.

Decision of the Falling Object

The falling objects (molecules, blockers, powerups) fall in specified numbers in a certain amount of time determined by the information of the game as described in the "building mode". The type of the object to fall is picked randomly each time.

Login Frame

The game always starts with the login frame. It asks the user to enter his/her username, to choose a saving option (file or database). The user sees two buttons either to start a new game or load game. After clicking the button "Start New Game", the building frame is opened to initialize the settings of the game. "Load Game" button does not work.

Building Mode

In building mode, the player specifies the initial settings of the game. In order to create a game environment, the user interacts with the system to specify the number of each game object (***atoms, reaction blockers, powerups, and molecules***). In addition, they specify the length unit L. For Alpha- and Beta-, there are two different molecular structures and the player can select one of the structures to appear in the game. Additionally, if the player selected a linear Alpha- or Beta-, they can choose these molecules to be stationary or spinning around their center while falling. The player can also specify the difficulty level of the game: easy, medium, and hard. If the level is easy then each object will fall in 1 second, in medium every ½ second, and in hard in every ¼ seconds. By default, the game will have the following numbers of objects:

- 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

Usability

Human Factors

The player will be able to see a monitor that will display the KUvid Game. Therefore:

- The text should be easily visible from 1 meter.
- Avoid colors associated with common forms of color blindness.

Reliability

Recoverability

If the game freezes there should be no need to restart the game and the game should continue from where it left off. The game is also savable, and can be paused if the user decides to resume the game, the game starts again from where it is paused. After saving, it is not possible to load the game because of some error in loading the game functionality.

Performance

The game should be played without any freezing of the screen or any crashes while playing the game which should not require the player to restart the game.

Supportability

Adaptability

The game should open in high quality graphical pixel representation in any type of local machine that is used. It should adapt the machine's window frame.

Free Open Source Components

In general, we used Java programming language and Eclipse as our IDE for coding and git to exchange and merge our codes for different tasks.

Interfaces

Noteworthy Hardware and Interfaces

- Screen monitor (this is perceived by operating systems as a regular monitor, for the mouse events)
- Keyboard
- Mouse

Glossary

Term	Definition and Information	Format	Validation Rules	Aliases
Shooter	A horizontally moving and rotating vehicle moved by the player via keyboard.			
Health Points	Health points can be reduced by the collision of the shooter and reaction blockers			
Score	Score can be increased by collisions of atoms with molecules, powerups with blockers and the score gained by each collision depends on the type of components colliding and shields of the atoms.			
Molecules	They are required to make a cure. They are initially at the top of the screen and fall either straight or zig-zag towards the bottom. There are 4 types of it; Alpha Beta Sigma, Gamma			
Atom	They are shot by the shooter to collide with the molecules. There are 4 types of it; Alpha Beta Sigma, Gamma			
Reaction Blockers	Each blocker can collide with a molecule and if so will make the molecule disappear. Also can harm the player if they fall on the ground by reducing the player's health points. There are 4 types of it; Alpha Beta Sigma, Gamma			
Powerups	They are used for destroying the blocker of the same type, they also fall from the top. They can be caught by shooting and colliding the atom of the same type. Then they are stored in the inventory and they can be shot by the shooter to collide and remove the corresponding blocker. There are 4 types of it; Alpha Beta Sigma, Gamma			
Blender	It is to be used to blend or break atoms and hence change the player's atom inventory.			
Shield	They are added to the atoms, they increase the speed and the efficiency (thus the score it will produce will be augmented) of the atoms. They have 4 types: Eta, Lota, Theta, Zeta			