INFO 3225 Individual Report

|  | Nathan Verma  100380776 |  |
| --- | --- | --- |

**Abstract**

I decided to attempt recreate the mobile game “Atomas”, where players are given a particle in the center of their screen. This particle can either be an element or a negative or positive atom. The player then can chose where they want to throw the atom into the ring the surrounds the center. The goal is to group similar atoms and merge them together by using charged atoms into a higher element depending on their atomic number (e.g. Hydrogen (H) + Hydrogen (H) = Helium (He)). Atoms keep appearing in the center until the outer ring is filled and you lose, you constantly try to get a higher score and avoid filling the ring.

A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated

Here is an example of how the game looks from the app. I ended up not implementing the full mechanics of the game such as the merging, and opted to go for a line instead of a ring. Many of the core mechanics of the game are still present in my project in which I will go over and explain, as well as some new features such as pressing 3 background particles in a limited time. The project uses many diffrenet mechanics and methods such; as ArrayList’s to hold particles in the scene, static arrays to hold element info, Multiple class files with heirchy such as Particle > ChargedParticle > NegativeParticle, some of these classes are abstract such as the Particle class, interfaces for display and movment, push and pop matrix in order to move particles, 5 different shape types includeing circle, line, squre, triangle and polygon, transformation and rotations with animations, multiple custom methods, keyboard and mouse interaction with particles, mouse hover mechanics, gui’s, and more.

**Use-Case Diagram**

The use diagram is as follows. There is a single actor that is the player. The player has options such as place particle, hover particle, press bg paricle or clear scene. If the player places a particle the score is increse. If the player hovers a element they can view element stats. If player presses the bg particle the set the particle to a negative to be placed. Finaly player can clear the scene

**A diagram of a flowchart

Description automatically generated**

**Class Diagram**

the following is a class diagram which shows the classes attribuests as well as their relations. The following can define the interactions. There are multiple types of particles, from elements to charged. There can be multiple of the same particle. Particles have location and size as well as particle specific attributes. Background particles exist and can be interacted with by changing motion and color. The game scene holds all elements as well as score. The elements array is used by the scene to create elements.

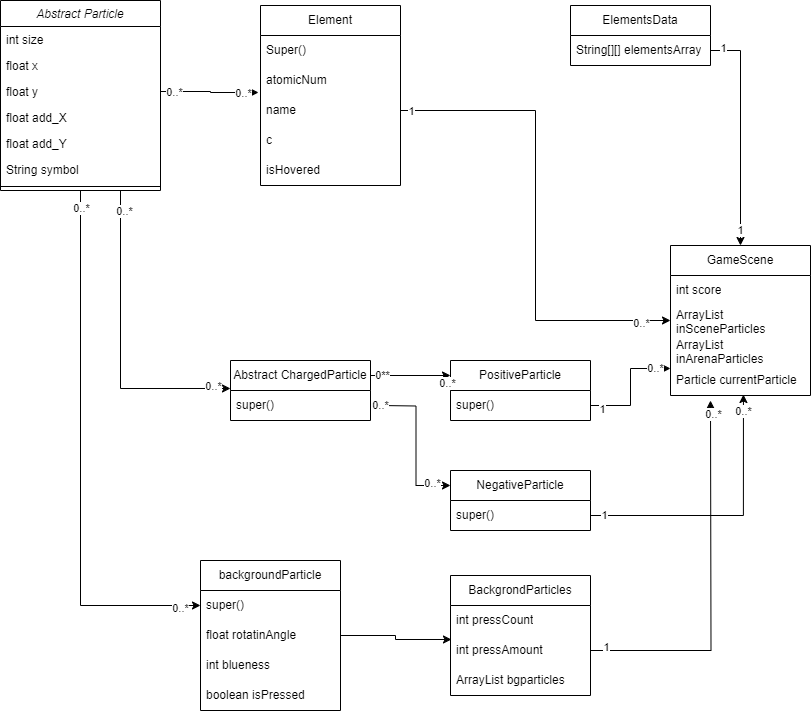
Abstract Particle: has essential data for all particles such as size, x, y, symbol

Element: has everything that particle has, but also atomicNum, name, c

positveParticle, negativeParticle: have the same attriputes as all particles but have a charge as a symbol ( -, +)

BackgroundParticles: can have motion and rotation as well as change in color and have a square as a symbol

GameScene: can hold the score of the game as well as all other particles in the areana



**Hiearchy and Particles**

All particles are based off the abstract Particle class which holds important info that all particles should have. Size is for the size of the particle, x for curent x axis location, y for current y axis location, add\_X and add\_Y for when the particle needs to be moved, and symbol which can either be an elemental symbol or a charge. Its methods include move(), to move the particle, shift() to translate the particle with an animation, getSymbol() to retrieve the symbol of the particle, and display() to draw the particle. Display is an interface which holds the display() method, and Moveable is also a interface which holds move() and shift() methods.

A screenshot of a computer program

Description automatically generated

Next we have Elements which are a child of Particle these have the same data as the normal Particle as it uses super() in the constructor to obtain the same attributes. Some aditions are atomicNum which holds the elements atomic number, the elements name, c for its color and a isHovered status to check if if the mouse is over it. This has different methods such as isMouseOver() to check if its being hovered, and displayElementalData() to display the elements stats on screen. It also has a modifed display() to suit its needs of displaying symbol and atomic number.

Next we have chargedParticles which is a child of Particle and a parent to negativeParticle and positiveParticle which we will get to later. This is mainly a parent class for the two charged particles to have similar characteristics. It also has a unique display() which draws charged particles as hexagons

A computer code with text

Description automatically generated

Next we have both negativeParticle and positiveParticle. These have uniqe symbols as “-” and “+” respectevly. These are displayed as hexagons due to the parent class.

Next we have BackgroundParticle which is a child of Particle and have a symbol of “■”. These are also unique as they have a roationAngle, bluneness (how blue they currently are) and and an isPressed status. This particle has unique methods that allow it to move and rotate such as moveRadomDirection() to allow random movment uitilizing move(), a changeMotion() when pressed, and a unique display() to show its quircky appearance of a rotating square with rotating triangles.

A screenshot of a computer program

Description automatically generated

Finaly we have the BackgroundParticles class to manage the background particles as well as give the player an oppurtinity to gain a negative particle. It has methods such as addParticle() that allow it to add background particles, an update() to display the particles, a resetParticles() to pereidocly reset the particles, and bgParticlePressed() to check if a particle has been pressed to change. bgParticlePressed() is interesting as it detects where the mouse is and if its close to a background particle that has not been pressed. It then calls the other methods such as ChangeMotion() to give it a new direction and speed. resetParticles() then removes all background particles and spawns new ones as to give the player a limited time to press 3 background particles to get a negatvie particle

A screenshot of a computer code

Description automatically generated

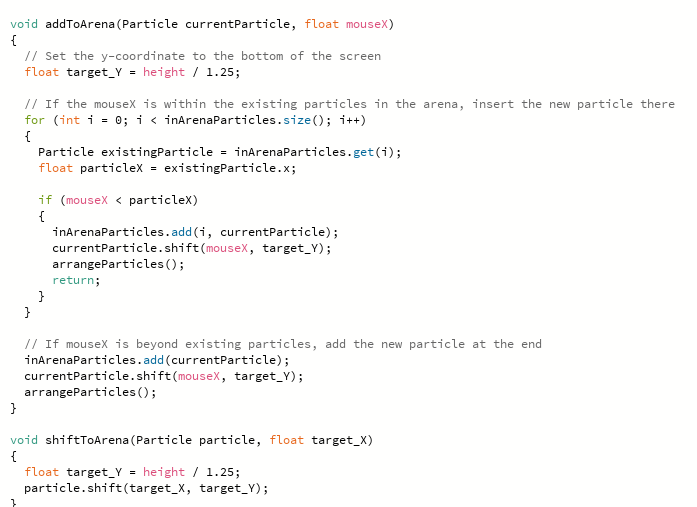
All together these make up the particle types that are seen in the project.

The Game Scene

GameScene class, This is where the particles are placed and where they are generated. I have multiple object arrays to hold particles depending on if they are in the entire scene or just the playable arena. Here we have methods such as addParticle() which when called, will allow you to add a particle into the scene array. Update() to show score, current particle, as well as a filter for if 3 background particles was pressed. generateRandomParticle which has a chance to call either generateRandomElement() or generateRadomChargedParticle(). generateRandomElement() will pick a random element from the ElementArray and depending on the score, you might get elements with higher atomic numbers. generateRadomChargedParticle functions simlarly with a chance to spawn wither a negative or positve particle. There is also GenerateAllElements() which adds every element to the scene to view. There is overrideCurrent() which is used when the player succesfuly presses 3 background particles and is rewared with a negative particle. There is clearInScene() which can be triggered to wipe all particles. There is addToArena() which will find where you clicked and move the particle to said space. There is shiftToArena() which will move the particle from the center to the line using shift(). There is a arrangeParticles() which will evenly space out the particles in the arena. There is score() which is used to add to the score. We also have printParticles() to print output to console and customFilter() which is used on 3 succesful background particle presses.

A computer screen shot of a code

Description automatically generated



**The Main Class**

The main class is simply where the methods are called and where the game is constanly updated. There is also text with directions to help the player. It also keepstrack of the backgroudn particle reset timer, and monitors for clicks and keyboard interacation from the user.

**The Gameplay**

Altough the merging feature is not implemented, it can be fun to look at all the different particles as well as fiddle with the back ground particles and try to catch 3 of them befor they reset. It also functions as a quick periodic table which is neat!

A screenshot of a computer game

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Reference:  
Sirnic. (2015). *Atomas*. Retrieved 2024, from https://play.google.com/store/apps/details?id=com.sirnic.atomas&hl=en\_US.

Atomas Wiki. Fandom. (n.d.). <https://atomas.fandom.com/wiki/Atomas_Wiki>

Welcome to processing!. Processing. (n.d.). https://processing.org/