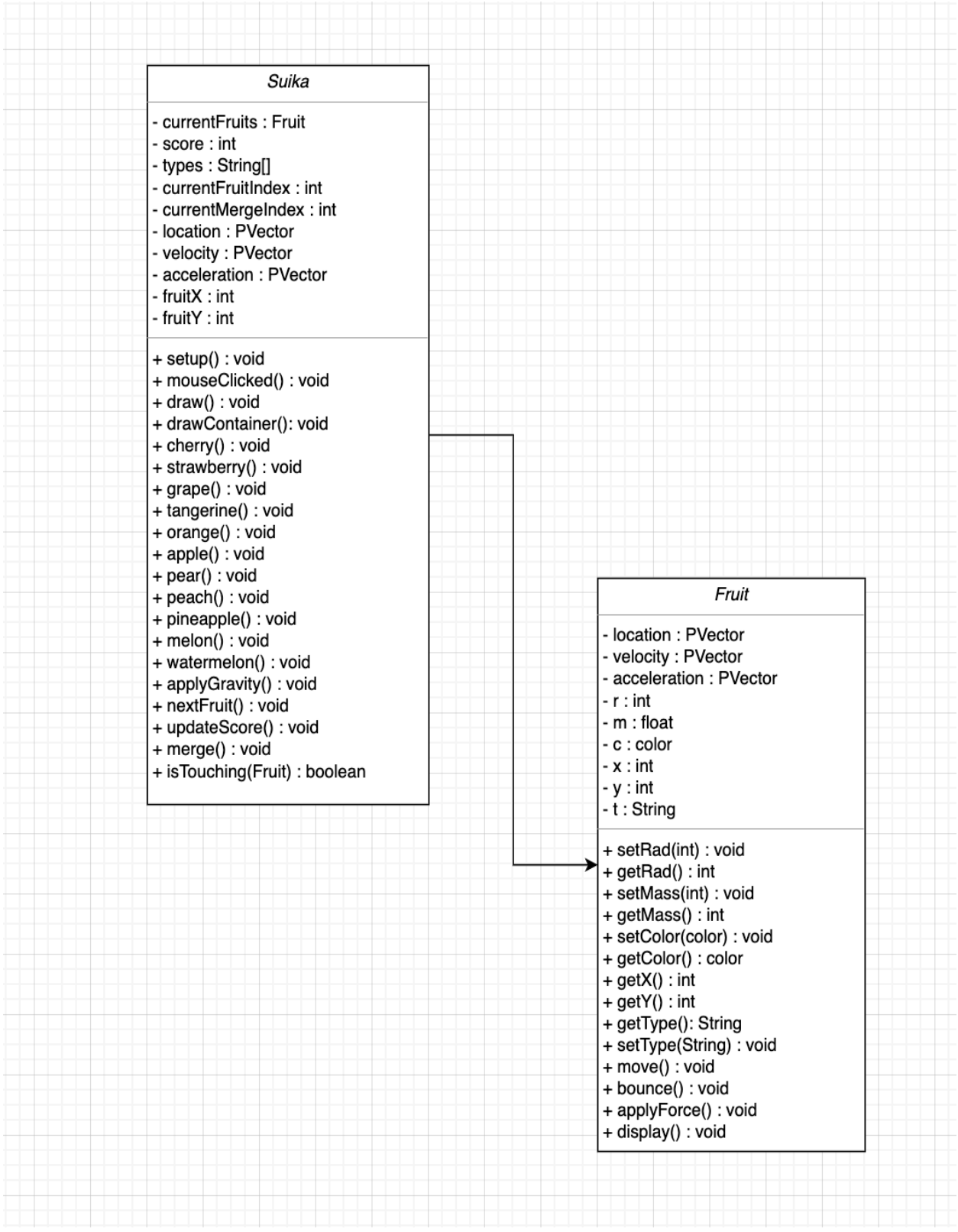
Period 6
Group Members: Jessica Yu and Ayesha Talukder
Group Name: Suika Fruits
Project Title: Suika Game

Description:

- Inspired by the actual Suika Game. You drop random fruit from the top of the screen into a container and when it touches something the same fruit as the fruit you dropped, it expands and turns into a slightly larger fruit type. It adds points whenever the fruit drops and points vary depending on the size of the fruit. Whenever fruits merge, the points added doubles. The game ends when no more fruit can be dropped into the container.
- Libraries: None needed so far
- Functionalities:
    - Gravity: to make the balls/fruits drop
    - Text: to record the points
    - Colors: to help distinguish different fruits

UML:

- Setup Class
- Draw Class


Suika
- currentFruits : Fruit
- score : int
- types : String[]
- currentFruitIndex : int
- currentMergeIndex : int
- location : PVector
- velocity : PVector
- acceleration : PVector
- fruitX : int
- fruitY : int
+ setup() : void
+ mouseClicked() : void
+ draw() : void
+ drawContainer(): void
+ cherry() : void
+ strawberry() : void
+ grape() : void
+ tangerine() : void
+ orange() : void
+ apple() : void
+ pear() : void
+ peach() : void
+ pineapple() : void
+ melon() : void
+ watermelon() : void
+ applyGravity() : void
+ nextFruit() : void
+ updateScore() : void
+ merge() : void
+ isTouching(Fruit) : boolean
Fruit
- location : PVector
- velocity : PVector
- acceleration : PVector
- r : int
- m : float
- c : color
- x : int
- y : int
- t : String
+ setRad(int) : void
+ getRad() : int
+ setMass(int) : void
+ getMass() : int
+ setColor(color) : void
+ getColor() : color
+ getX() : int
+ getY() : int
+ getType(): String
+ setType(String) : void
+ move() : void
+ bounce() : void
+ applyForce() : void
+ display() : void

4. How it works

- Step 1:
    - Drop a fruit at the desired position (using the mouse to activate)
    - If fruit touches similar fruit, the fruits will merge
    - else the fruits will stay in respective places

  Step 2:
    - Repeat until the container can no longer hold any more fruit
    - It returns the number of points you got

Objective: get the most amount of points possible without completely filling in the container.
Keys To Use:

- Click left button on mouse to drop
- Move mouse cursor along the x axis to drop fruit in different parts of the container

5. Functionalities / Issues

- Current Functualities:
    - Direction - can move the fruit from the starting position side to side using mouse cursor and drop by clicking left button of mouse
    - Draw - calls drawContainer, drawFruit, and updateScore methods - draws the initial container, respawns a new fruit every time a fruit drops, and keeps track of the points based on fruits dropped/merged
    - Different Fruits methods - called by nextFruit method to spawn different fruits
    - UpdateScores - score gets updated differently based on the currentFruitIndex, which keeps track of which fruit nextFruitIndex is looking at
    - ApplyGravity Method - using PVector, drops the ball to the bottom of the container when mouse is clicked
- Functionalities Planned to be done by next meeting
    - Merge Method - using updateScore and drawFruit methods
    - isTouching Method - prevent the fruits from overlapping
- Issues During Development
    - We had issues with figuring out how to move the fruit on the top, but eventually we opted on using keyboard controls over mouse controls.
    - We didn’t know how we were going to draw the fruit and store their variables. To fix this issue, we made a fruit constructor and assigned each fruit to a different method that created a new fruit object using the constructor.

- We had issues with swapping between different fruits spawning. We fixed this by creating a new method that kept track of different fruits in a fruit array that we created, containing the names of all the fruits. Everytime the nextFruitIndex method is called, it adds one to the currentIndex and spawns whatever fruit is associated with that index number.
- We had a huge issue with overlapping fruit because we couldn't figure out how to identify whether there was an object underneath the fruit that just got dropped

6. Log
    - Ayesha
        - Helped with implementing many of the game methods in the Suika class
            - Fruit Constructor
            - Direction Method
            - Draw Methods
            - Update Score Method
        - Developed the Fruit Class
        - Helped Develop Force on the Balls and got the Fruits to drop
    - Jessica
        - Helped brainstorm the game idea and the different methods that would be used in executing the game
            - updateScore, merge, gravity, draw, etc.
        - The different fruit drawing methods
            - Cherry(), Strawberry(), Grape(), etc.
        - Altered methods and constructor in the fruit class
        - mouseClicked method
        - nextFruit method
        - isTouching method