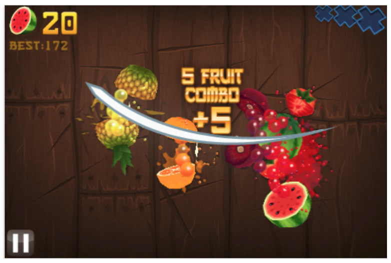
Period 9 APCS
Kevin Zhou and Zhi Ming Xu
Group Name: Code Ninjas

Brief Project Description:



Fruit Ninja is a classic mobile arcade style game involving slicing fruit that is tossed up on the screen in order to increase their score. The ‘run’ is over once a certain amount of lives are depleted as a result of too many fruits being missed by the user. As the score progresses, the velocity and frequency of the fruit tossed up gradually increases at a fixed rate.

In our MVP, we will be implementing some of the basic features of Fruit Ninja, including the classic arcade mode, combos, lives, and progressive difficulty as well as the mechanics involving the generation, directional trajectory, and rotational trajectory of the fruit as well as the slicing mechanism.

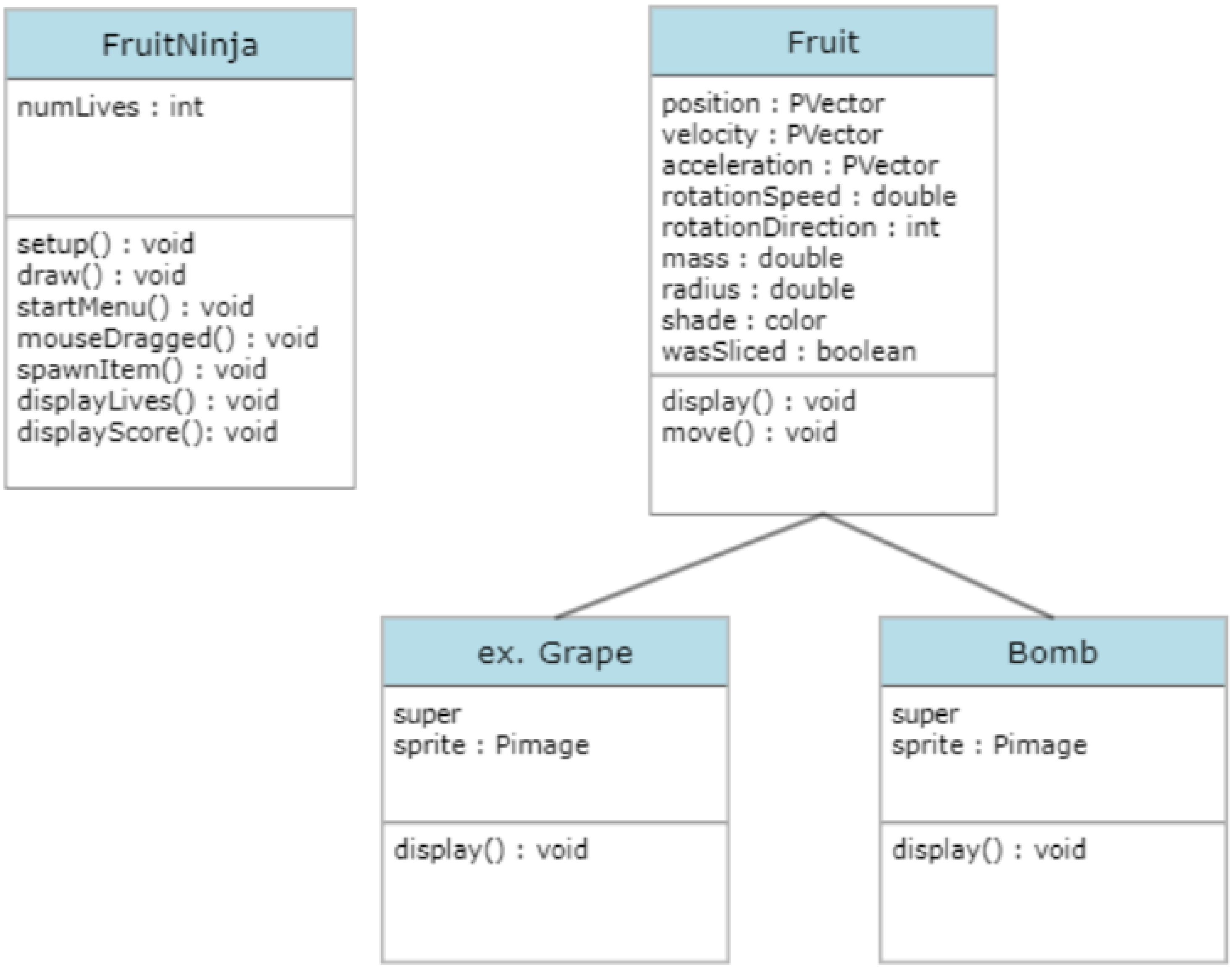
Minimum Viable Product:

- One type of fruit
- One type of bomb
- Implementation of Gravity and Spinning Fruit
- Fruit can be sliced
- Fruit halves maintains velocity and spin after being sliced
- Score/Combos
- Lives
- Progressive Difficulty
- 1 gamemode- arcade mode
- Menu interface
- Fruit will not be 3 dimensional

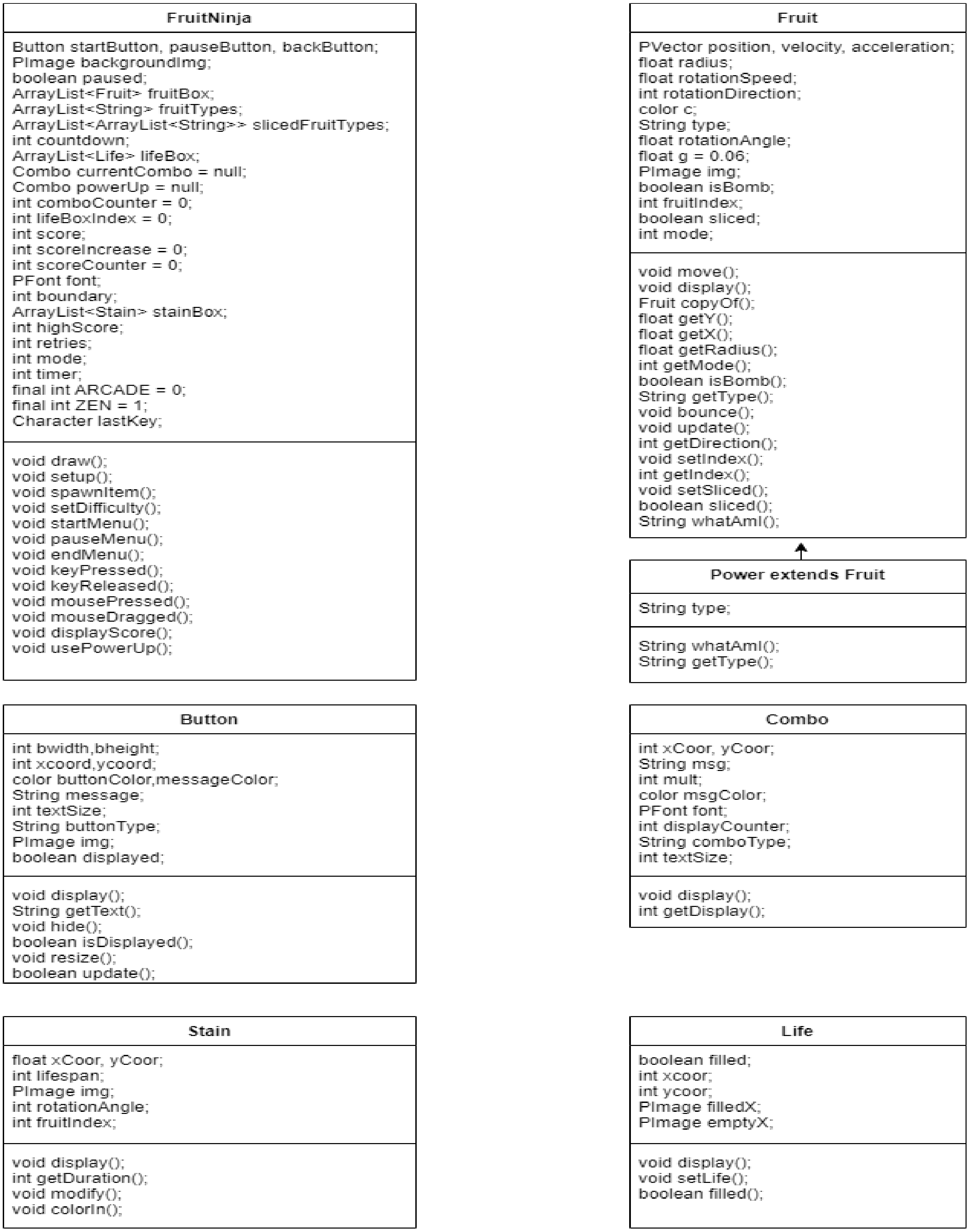
Nice to Have Features:

- Additional types of fruit
- Visual effects when fruits are sliced
- Power ups
- Additional game modes
- Combo multipliers when multiple fruits are sliced at the same time
- Displaying the highest score after multiple runs

BEFORE:


FruitNinja
numLives : int
setup() : void
draw() : void
startMenu() : void
mouseDragged() : void
spawnItem() : void
displayLives() : void
displayScore(): void
Fruit
position : PVector
velocity : PVector
acceleration : PVector
rotationSpeed : double
rotationDirection : int
mass : double
radius : double
shade : color
wasSliced : boolean
display() : void
move() : void
ex. Grape
super
sprite : Pimage
display() : void
Bomb
super
sprite : Pimage
display() : void

AFTER:



```
FruitNinja
Button startButton, pauseButton, backButton;
PImage backgroundImg;
boolean paused;
ArrayList<Fruit> fruitBox;
ArrayList<String> fruitTypes;
ArrayList<ArrayList<String>> slicedFruitTypes;
int countdown;
ArrayList<Life> lifeBox;
Combo currentCombo = null;
Combo powerUp = null;
int comboCounter = 0;
int lifeBoxIndex = 0;
int score;
int scoreIncrease = 0;
int scoreCounter = 0;
PFont font;
int boundary;
ArrayList<Stain> stainBox;
int highScore;
int retries;
int mode;
int timer;
final int ARCADE = 0;
final int ZEN = 1;
Character lastKey;

void draw();
void setup();
void spawnItem();
void setDifficulty();
void startMenu();
void pauseMenu();
void endMenu();
void keyPressed();
void keyReleased();
void mousePressed();
void mouseDragged();
void displayScore();
void usePowerUp();
```

```
Fruit
PVector position, velocity, acceleration;
float radius;
float rotationSpeed;
int rotationDirection;
color c;
String type;
float rotationAngle;
float g = 0.06;
PImage img;
boolean isBomb;
int fruitIndex;
boolean sliced;
int mode;

void move();
void display();
Fruit copyOf();
float getY();
float getX();
float getRadius();
int getMode();
boolean isBomb();
String getType();
void bounce();
void update();
int getDirection();
void setIndex();
int getIndex();
void setSliced();
boolean sliced();
String whatAmI();
```

```
Power extends Fruit
String type;

String whatAmI();
String getType();
```

```
Button
int bwidth,bheight;
int xcoord,ycoord;
color buttonColor,messageColor;
String message;
int textSize;
String buttonType;
PImage img;
boolean displayed;

void display();
String getText();
void hide();
boolean isDisplayed();
void resize();
boolean update();
```

```
Combo
int xCoor, yCoor;
String msg;
int mult;
color msgColor;
PFont font;
int displayCounter;
String comboType;
int textSize;

void display();
int getDisplay();
```

```
Stain
float xCoor, yCoor;
int lifespan;
PImage img;
int rotationAngle;
int fruitIndex;

void display();
int getDuration();
void modify();
void colorIn();
```

```
Life
boolean filled;
int xcoor;
int ycoor;
PImage filledX;
PImage emptyX;

void display();
void setLife();
boolean filled();
```

Phase 1: Creating the startMenu() Method
- Write a method that properly initializes the screen to the start menu
- The method is called in the setup() function
  - The start menu should have a "Start" button~~ and a "How to Play" button~~
  - Implement a Button Class
  - The arena should have a pause button

Phase 2: Implement the Fruit Class
- The Fruit Class has 3 PVector fields for position, velocity, and acceleration, a double field to represent rotational speed, and an int field to represent rotational direction, and double fields to represent radius, mass, and a field for color as well as a boolean to represent if the fruit has already been sliced
- Temporary Implementation: Fruits are colored ellipses to confirm that rotation works before importing fruit images.
- Fruits will be spinning at a constant velocity while being tossed up/down.
- ~~Other fruits that will be added later on will be subclasses of the Fruit class~~

Phase 3: Implementing Slicing Method
- Will be written inside the mouseDragged() method in processing
- Temporary Slicing Detection: The mouse coordinates are within the area of a fruit for a constant amount of time (Temporary Implementation)

Phase 4: Implement Bomb ~~Class~~ Field in Fruit Class
- ~~Will be inherited from the Fruit class to obtain the physics of fruit motion~~
- Utilizes the slicing method in phase 3 to automatically end the current run when sliced.

Phase 5: Implement spawnItem Method
- ~~Will be implemented within the draw() method~~
- Spawn fruit at a constant rate throughout a run (Temporary Implementation)
- Fruit is spawned at a constant y value that is below the screen, a random x value such that the fruit spawns completely within the screen

Phase 6: Implementing Lives~~ Field~~ Class
- This will be a global ~~field~~ class that will be modified every time a fruit falls without being sliced. It will be displayed near the top right corner of the screen.
- Once the number of lives reaches 0, the current run will end, then the startMenu() method will be called

Phase 7: Implement Progressive Difficulty
- Time is tracked using the "frames" variable in processing
- As time goes on, both the spawn frequency of the fruits and the velocity of the fruits will be increased to make it harder for the user to maintain all their lives.

Phase 8: Implement Score + Combos

- The score will be maintained as a global variable that gets increased every time a fruit is successfully sliced. It is displayed using the text() function at the top of the screen.
- If a certain amount of fruit is sliced within a constant amount of time, the player is granted score bonuses dependent on the number of fruit sliced (Temporary implementation)

Phase 9: Implement Sliced Fruit field

- Two sliced fruit will spawn for each fruit that is sliced
- Is a field within the Fruit Class

Phase 9: Implement Stain Class

- Stains will be spawned when a fruit is sliced and will fade away after a constant amount of frames
- Stains will be rotated by a random amount and have a color depending on what fruit caused the stain to spawn in.

Phase 10: Implement zen mode

- Zen mode is another gamemode that, instead of lives and bombs, has a set amount of time and power-ups
- Replace start button in the menu with fruits that can be sliced to choose a mode

Phase 11: Implement Power Class

- Power is a subclass of Fruit. When sliced, Power fruits provide special effects:
- Frenzy power up tosses up 10 fruits in a horizontal line for an easy 10-combo
- Bonus power up doubles the score

Phase 12: Bug testing + submission

- Fix various glitches and bugs, update dev log accordingly