Documentation

STATEMENT

Enfloral draws heavy inspiration from classic point-and-click adventure games, combined with mild puzzle elements, noir dramas, and dashed with colour palletes that are a crossbreed between vintage and neon.

Enfloral is artistic exploration to its core, but I did experiment with a lot of ideas taught in lecture and lab, such as OOP to structure the game's flow and its object interactions, PVectors to arrange props and create the ecosystem in the Tank, and a few uses of sin and cos to add further animation.

Below you'll find documentation and rough code/screenshots compiled throughout my entire work period; 3 days of Enfloral coding.

You can watch the demo and timelapses here:

https://youtu.be/g7S_itNT2Xw

https://youtu.be/V5nWPDR7OH0

https://youtu.be/BGnmAGCVioY

DAY 1

- Drew character designs
- Set up sketch
  - gameMode function
  - Clock that works

The character designs were inspired by GDYU's (Game Devs at York U) *Create-Tober* monthly challenge. The challenge offers a colour pallete for every week of October, and so these characters were created with said pallete! (And also because I am the organiser of this event).

DAY 2

- Setting up buttons to trigger gameMode string change, used OOP

Enfloral.pde

```
String gameMode = "";
Buttons button;
```

buttons.pde

```
class Buttons {
  PVector pos;
```

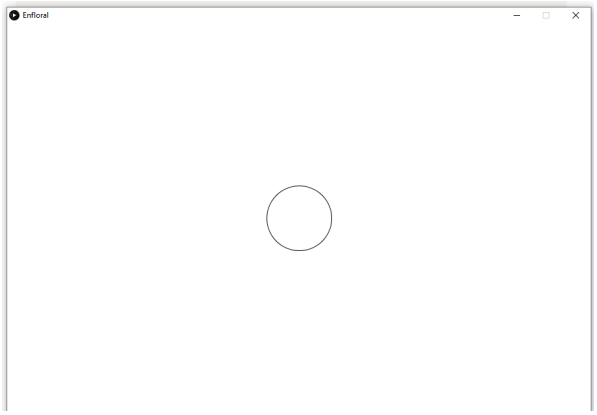
```
void setup() {
 size (900, 600);
 button = new Buttons(new PVector(width/
2, height/2), 100, "clockRoom");
}

void draw() {
  background(255);
  if (gameMode == "clockRoom") {
   clockRoom();
  } else if (gameMode == "reset") {
   background(255);
  }

 button.run();

}

void keyPressed() {
 if (key == '1') {
  gameMode = "clockRoom";
 } else if (key == '2') {
  gameMode = "reset";
 }
}
```

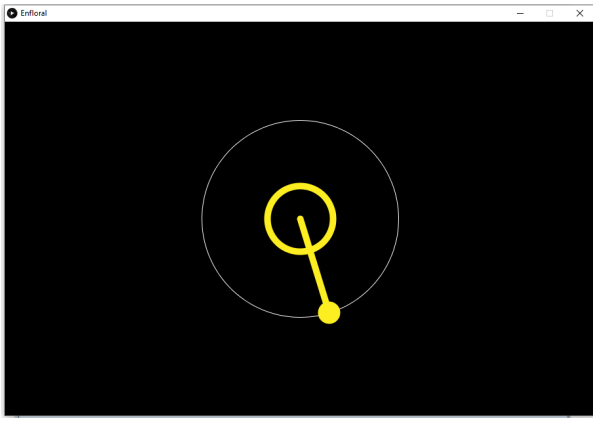
```
  int size;
  String id;

  Buttons (PVector pos, int size, String
id) {
    this.pos = pos;
    this.size = size;
    this.id = id;
  }

  void drawButton() {
    ellipse(pos.x, pos.y, size, size);
  }
  void buttonClicked() {

    if (dist(mouseX, mouseY, pos.x, pos.
y) <= size/2) {

      if (mousePressed) {
        gameMode = id;
      }
    }
  }


  void run() {
    buttonClicked();
    drawButton();
  }
}
```

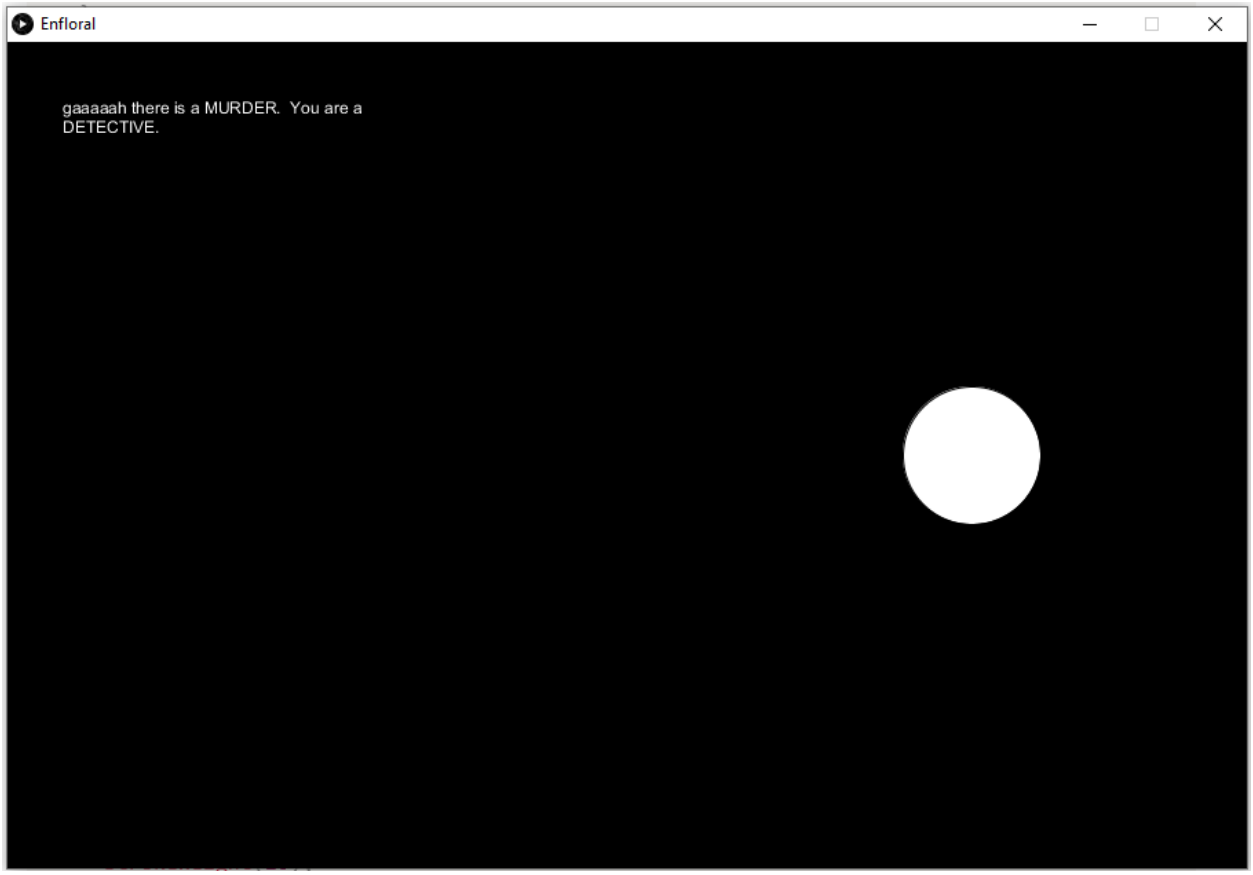
clock.pde

```
float angle;
boolean magnetGone = true;
void clockRoom() {
   int s = second();

  background(0);
  stroke(255);
  noFill();
  pushMatrix();
  translate(width/2, height/2);
  strokeWeight(1);
  float rad = 150;
  circle(0,0,rad*2);

  strokeWeight(34);
  stroke(252,238,33);
  float x = rad*cos(angle);
```

```
  float y = rad*sin(angle);
  point(x,y);
  strokeWeight(10);
  line(0, 0, x, y);
  popMatrix();

  if (magnetGone) {
  if (s%2==0) angle += 0.005;
  }
}
```



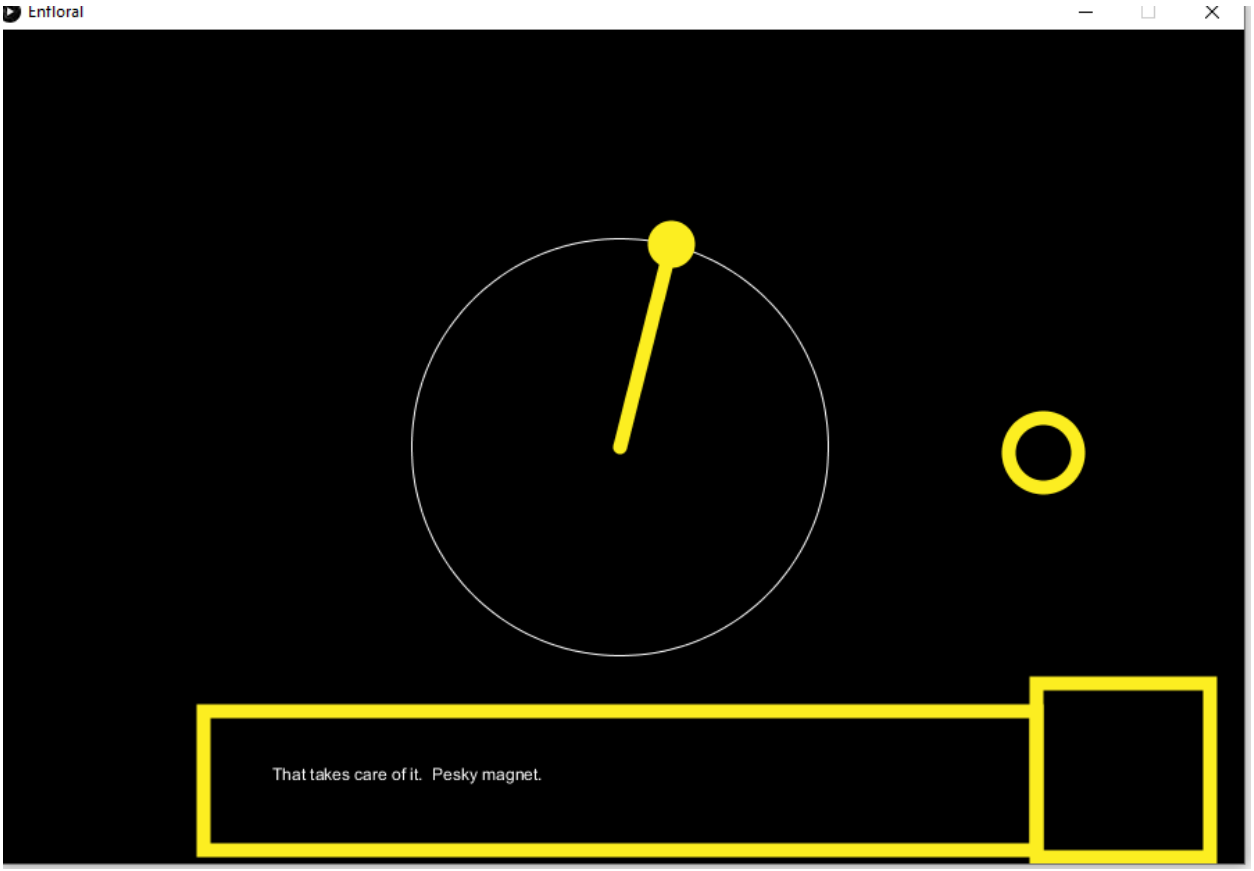




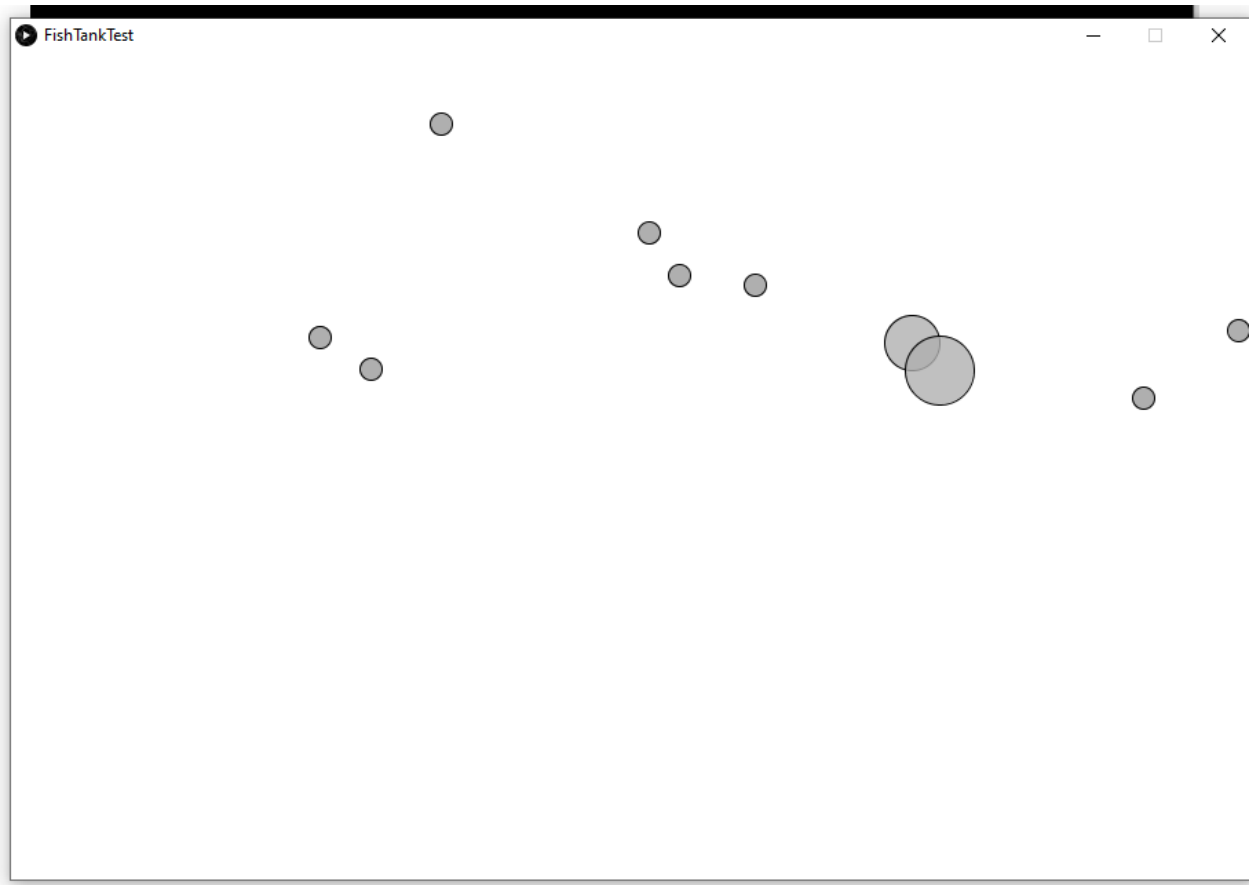
- Implemented UI elements, text change with setter methods. The item in the item box area is draggable, with padding to keep distance consistent when moving item around fast

```
PVector textBoxPos, itemBoxPos;
PVector itemPos = new PVector(810, 530);
String desc = "beginning";
String item;
int padding = 50;
int itemSize = 50;

class UI {
  void drawBoxes() {
    textBoxPos = new PVector(150, 490);
    rect(textBoxPos.x, textBoxPos.y, 600, 100);
    itemBoxPos = new PVector(750, 470);
    rect(itemBoxPos.x, itemBoxPos.y, 125, 125);
    text(desc, textBoxPos.x+padding, textBoxPos.y+padding);
  }

  void drawItem() {
```

```
    ellipse(itemPos.x, itemPos.y, itemSize, itemSize);
  }
  void setItem(String newItem) {
    item = newItem;
  }

  void setDesc (String newDesc) {
    desc = newDesc;
  }

  void dragItem() {
    if (mousePressed) {
      if (dist(mouseX, mouseY, itemPos.x, itemPos.y) <= itemSize + padding) {
        itemPos.x = mouseX;
        itemPos.y = mouseY;
      }
    } else {
      itemPos.x = 810;
      itemPos.y = 530;
    }
  }

  void run() {
    drawBoxes();
    drawItem();
    dragItem();
  }
}
```



Built an Mover and Attractor test to simulate pulling something up with a magnet and attracting Movers with it



DAY 3

Worked on implementation of art assets for the UI

Added to placeholders for interactive objects. Player can fix clock by removing the magnet on the dot of the clock arm. Then, use the magnet to fish out the pin in the Tank. Variables set in place— setter methods used— to keep track of which item ID is in the inventory box.