Exercise 07: Progress Report

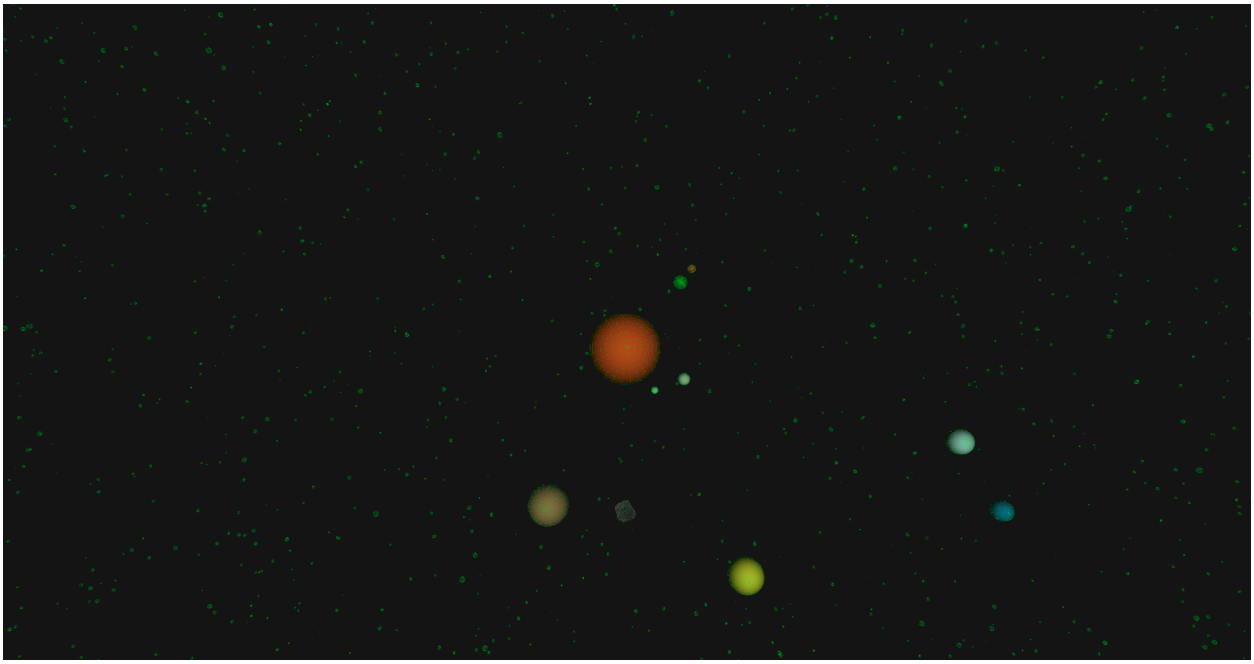
Class	Creative Computation I
Created	@Nov 16, 2020 11:53 PM
Key Words	Final Project / Progress Report
User	Melissa Banoen-Garde
Week	Week 11

Stars

Progress

I created a Star class object with a `motion()`, `display()`, and `checkStar()` method. A total of 2000 stars are stored in an array and displayed as tiny spheres floating on the X, Y, and Z axes. I added a "z" parameter and randomized its range between -1000 and 200. This was exciting (I can't even describe the feeling of overthinking only to find out that it's quite simple, "exciting" is an understatement) to see it work because I've been pondering how to "talk about depth" or describe depth in Javascript. We link X with "width", and Y with "height" but "depth" isn't programmed to correspond with Z. For instance, I can't type

`random(-depth, depth);`. I tweaked a code provided in the week we learned about arrays. The tweak allows the stars to float and change directions more fluidly in a space-like manner.



Screenshot of stars displayed in the simulation

Challenges faced

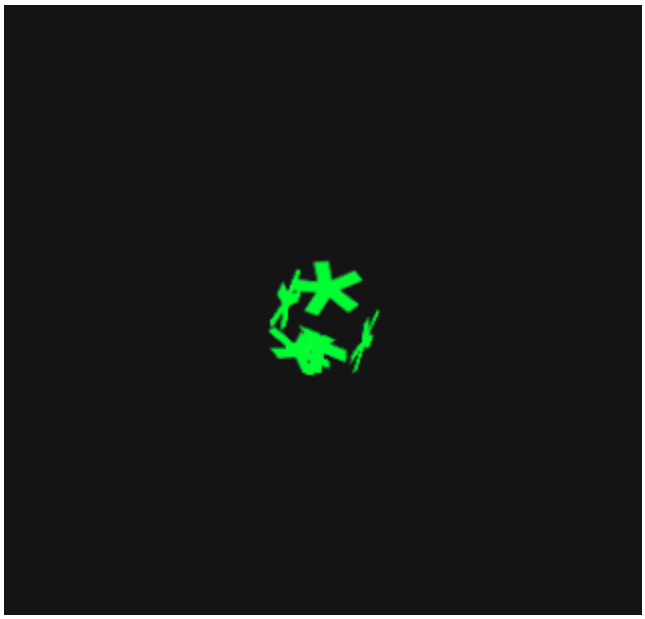
- Initially, I wanted the stars to be a for-loop of the star symbol "*" using `text()`. Unfortunately, this did not work at the first try.
- My second attempt was by creating a "star" variable in the display method of the Star class object and then defining it with a `createGraphics()` function (like in p5's example).
- Using `createGraphics()`, I wanted to "create" a new texture using the star symbol centered on each sides of the a box (cube).
- This, however, failed to multiply and rendered the simulation much *much* slower.

```
class Star {

  constructor() {
[...]
```

```
function setup() {
  createCanvas(400, 400, WEBGL);
}
```

```
    this.string = `x`,
    this.centerX = 25,
    this.centerY = 25,
    this.textSize = 15,
    this.textX = 25,
    this.textY = 25,
    this.boxSize = 50
  };

display() {
    push();
    let star = createGraphics(5
0,50);
    star.fill(this.colour.r, thi
s.colour.g, this.colour.b, this.
colour.alpha);
    star.textAlign(CENTER, CENTE
R);
    star.textSize(this.textSiz
e);
    star.text(this.string, this.
textX, this.textY);

    texture(star);
    noStroke();
    box(this.boxSize, this.boxSi
ze, this.boxSize);
    pop();
  }
```

```
function draw() {
  background(20);
  noStroke();

  push();
  rotateX(frameCount * 0.01);
  rotateY(frameCount * 0.005);

  let star = createGraphics(100, 100);
  star.fill(0,255,0);
  star.textSize(250);
  star.textAlign(CENTER, CENTER);
  star.text("*", 50, 120);
  texture(star);
  box(50);
  pop();
}
```



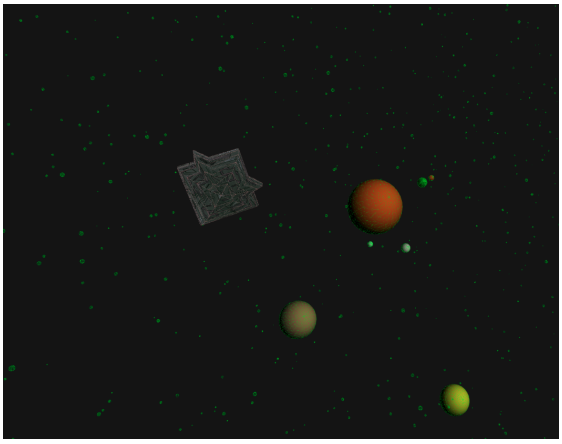
Individual star 1.2

- In Exercise 06, the synth had no issues being played once the stars are eaten. However, when trying to implement what was programmed from Exercise 06 into my final project, the program stops and all my stars disappear. In order to move on to other elements, I removed the synth (for now) replaced it with a preloaded bell sound effect.

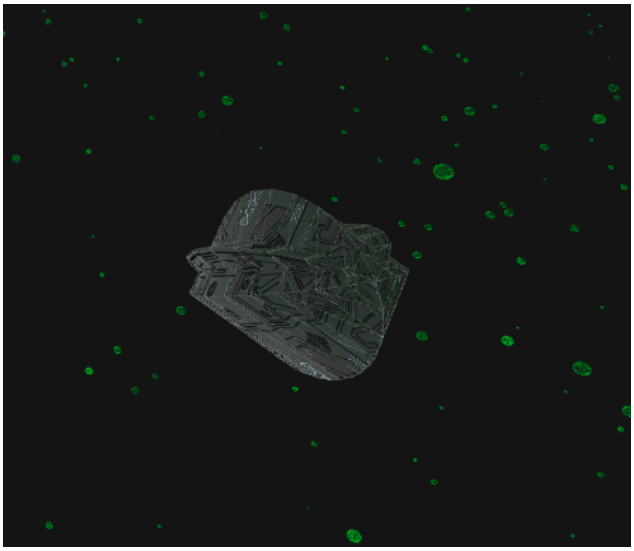
Spaceship

Progress

- I created a "User.js" class object which is displayed as a spaceship in the simulation. The class object has a `motion()` and and `display()` method. Its parameters are `x`, `y`, `z`, `size`, and `spaceshipTexture`.
- In `motion()`, I added the user's handle input using the keys AWSD and QE. The additional keys "Q" and "E" allows the user to move within the solar system's depths (z-axis).
- In `display()`, the spaceship is composed of two spheres with custom subdivisions in their x and y dimension with a sci-fi texture found on Pinterest.



Spaceship in motion



A closer look at the spaceship's appearance

Challenges faced

- One thing I've been having difficulties with, in general, was calling the synth or oscillator in a class' constructor. In the spaceship's case, I called the oscillator in its class constructor and created a `playNote()` method so it may be triggered when the the handle inputs are pressed. Sound works but very distorted. So I tried to to call the oscillator in the main script and gave it a variable to add within the parameters of the User's class object. This too failed. Instead, for now, I called the oscillator in the script and created a `keyPressed()` and `keyReleased()` function, giving the illusion that the spaceship makes noise each time it moves.

Camera

Progress

- This element follows the spaceship and can be controlled by the user, using their mouse or mousepad.
- I declared three variables in the script; `camX`, `camY`, and `camZ` with a value of 0. Under the `motion()` method of the user's class object and handle inputs, I defined the cams' positions to correspond with the user class object's speed.

```
// User's movement
  motion() {
    this.x += this.vx;
    this.y += this.vy;
    this.z += this.vz;

  // Handle input and direction
  if (keyIsDown(65)) {        // left
    this.vx = -this.speed;
    camX -= this.speed;
  }
  else if (keyIsDown(68)) {   // right
    this.vx = this.speed;
    camX += this.speed;
  }
  else {
    this.vx = 0;
  }

  if (keyIsDown(87)) {        // up
```

```
    this.vy = -this.speed;
    camY -= this.speed;
  }
  else if (keyIsDown(83)) {    // down
    this.vy = this.speed;
    camY += this.speed;
  }
  else {
    this.vy = 0;
   }

   // Depth function for user to move on z-axis by pressing "o" and "l"
   if (keyIsDown(69)) {       // forward E
     this.vz = -this.speed;
     camZ -= this.speed;
   }
   else if (keyIsDown(81)) {  // backwards Q
     this.vz = this.speed;
     camZ += this.speed;
   }
   else {
     this.vz = 0;
   }
  }
```

- I added the three variables in the simulation's `camera()` position: `camera(camX, camY, (height/2) / tan(PI * 30 / 180) + camZ, camX, camY, 0, 0, 1, 0);`
- For the user to control the camera's view, I defined two variables; `mousecamXmap` and `mousecamYmap.` Both variables remaps the mouseX's and mouseY's range. They are also added in the camera's parameters.

```
camera(camX, camY, (height/2) / tan(PI * 30 / 180) + camZ, camX + mousecamXmap, c
amY + mousecamYmap, mousecamXmap+mousecamYmap, 0, 1, 0);
```

Challenges faced

- I had the most difficulties trying to find the most simple yet effective way to have the camera function feel natural in both its trailing and user-control. Manipulating the camera with the mouse doesn't pan as smooth as I'd want it to and sometimes I find myself trying to recalibrate the the camera to view the spaceship. You lose the spaceship sometimes.

- I struggled with understanding p5's example of the `mouseWheel()` function. This is something I will return to because I want the user to be able to zoom closer to the simulation's spaceship. I feel that this would facilitate the user's quest at collecting stars.

Planet Subclasses

Progress

- I gave each planet a subclass that inherits from the `Planet.js` superclass object. They are declared with new variables in the main script, and push and stored in the "planets" array.
- I created a for-loop that runs the Planet.js superclass' methods into each value in the planets array (polymorphism!).

Challenges faced

- I learned that order is extremely important when programming with subclasses. This was made clear upon an office hours meeting with Pippin when my planets were not displaying. *The rule of thumb is to keep translates and rotates within the push and pops of a 3D primitive, in WEBGL*. Giving each planet's subclass a custom colour (for the sake of seeing the outcomes of what I'm programming, it can be hard to see when everything is green) was successful but, oddly, I cannot display saturn's rings. I've tried programming a new `torus()` within a push and pop function yet nothing appears. I placed the `torus()` above and below the `super.display()` function, with and without the push and pops and, still, nothing appears.

Future work

1. First state: intro + instructions of what to collect in order to unlock a planet
2. Stars' appearance will appear as mini spheres with different colours referring to its respective planet.

3. Only two planets (plus the Sun) are active. User must click them for infos. User must accumulate stars of respective planets to 'unlock' its visibility and have access to its infos. I will add an `infoDisplay()` method in the superclass `Planet.js` and a `this.active = false` instruction, under the constructor, so that the planets that aren't supposed to appear at the beginning aren't active.
4. Once enough stars have been accumulated, a 'you've unlocked ____' sound (and maybe notification box?) is triggered to appear on screen.
5. User must avoid asteroids or they may lose a portion of the stars they've collected. Will figure out how to incorporate this.