Snake Game part 6

Step 6: Eat Food

Now that we have our snake and the food, we need our snake to be able to eat the food, and grow after eating. Let's code that!

Because we are working with a grid system, we don't need to use complicated collision detection. Instead, as long as our snake's head is in the same grid square as the food, it will eat it and grow.

Let's make a method to allow the snake to eat!

```
void eatFood(PVector food) {
  if (snake.x == food.x && snake.y == food.y) {
    tailLength++;
    newFood();
  }
}
```

This method takes the PVector object, which will contain the x and y location of the food. It then compares this against the snake's x and y location. If they are the same, the snake gains a tail, and new food is placed randomly on the board.

(Also, while doing this, I realized that we have a little bit of a bug. We need to move the tail, before moving the snake's head, or else we will be missing the first tail section.)

```
void draw() {
  background(0);
  showFood();
  eatFood(food);
  moveTail();
  moveSnake();
  showSnake();
  showTail();
}
```

Your code will now look like this.

```
int gridSize = 20;

PVector snake;
int snakeSize;
int xSpeed;
int ySpeed;
int tailLength;
ArrayList<PVector> tail = new ArrayList();
```

```
PVector food;

void setup() {
  size(600, 600);
  frameRate(10);
  newGame();
}

void draw() {
  background(0);
  showFood();
  eatFood(food);
  moveTail();
  moveSnake();
  showSnake();
  showTail();
}

void newGame() {
  snake = new PVector(width/2, height/2);
  snakeSize = gridSize;
  direction(1, 0);
  newFood();
}

void showSnake() {
  fill(100, 215, 0);
  square(snake.x, snake.y, snakeSize);
}

void moveSnake() {
  snake.x += xSpeed;
  snake.y += ySpeed;

  // This keeps the snake on the board
  snake.x = constrain(snake.x, 0, width-gridSize);
  snake.y = constrain(snake.y, 0, height-gridSize);
}

void moveTail() {
  if (tailLength > 0) {
    if (tailLength == tail.size() && !tail.isEmpty()) {
      tail.remove(0);
    }
    tail.add(new PVector(snake.x, snake.y));
  }
}

void showTail() {
  for (PVector section : tail) {
    square(section.x, section.y, gridSize);
```

```
  }
}

void direction(int x, int y) {
  xSpeed = x * gridSize;
  ySpeed = y * gridSize;
}

void keyPressed() {
  if (keyCode == UP) {
    direction(0, -1);
  } else if (keyCode == DOWN) {
    direction(0, 1);
  } else if (keyCode == RIGHT) {
    direction(1, 0);
  } else if (keyCode == LEFT) {
    direction(-1, 0);
  }
}

void newFood() {
  int columns = width/gridSize;
  int rows = height/gridSize;
  int randomColumn = (int)random(columns) * gridSize;
  int randomRow = (int)random(rows) * gridSize;
  food = new PVector(randomColumn, randomRow);
}

void showFood() {
  fill(255, 0, 100);
  square(food.x, food.y, gridSize);
}

void eatFood(PVector food) {
  if (snake.x == food.x && snake.y == food.y) {
    tailLength++;
    newFood();
  }
}
```

Challenges

1. Add a score.
2. Make a bonus food worth more points appear at random intervals.