Section A
Unit #10
Vectors

Content

Section A
Unit #10 Vectors

Unit #10 Vectors

Vectors may sound like maths you might want to forget but if you are going to draw things, move them and manipulate them then this is going to be critical. Vectors aren't as challenging as they may look initially (assuming you have little understanding or recollection of them) and even though they are a little bit abstract they have real world applications.

Creates a new p5.Vector (the datatype for storing vectors). This provides a two or three dimensional vector, specifically a Euclidean (also known as geometric) vector. A vector is an entity that has both magnitude and direction. As you go through the sketches you will see how this works.

It uses the **createVector(x, y)** function

Sketch A10.1 a bouncing ball

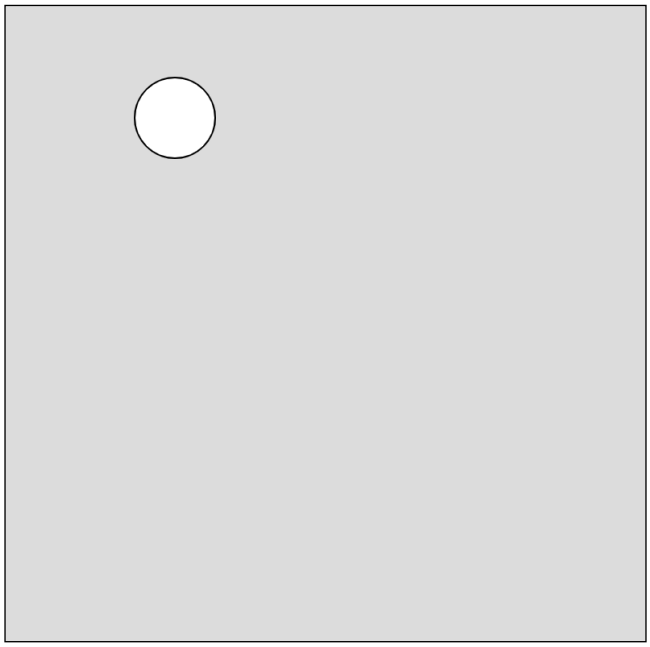
We give a circle an x velocity and a y velocity. The **if()** statements check to see if the ball has reached any of the edges of the canvas. Make sure you understand what each variable is doing before going onto the next Sketch Awhere we introduce vectors. The variables vel_X and vel_Y are short for velocity x and velocity y.

Introducing ***=** which is similar to **+=** but rather than just adding the value it multiplies it instead

```
let x = 100
let y = 100
let vel_X = 2.5
let vel_Y = 4

function setup()
{
  createCanvas(400, 400)
}

function draw()
{
  background(220)
  x += vel_X
  y += vel_Y
  if ((x > width) || (x < 0))
  {
    vel_X *= -1
  }
  if ((y > height) || (y < 0))
  {
    vel_Y *= -1
  }
```

```
  circle(x, y, 50)
}
```

Ball bouncing off sides



Notes

This is to demonstrate how we would normally make a ball bounce round the canvas against the edges without using vectors, that is coming next. You can use underscore vel_X in your variable naming process, this is not uncommon.

Challenge

Add another ball

Sketch A10.2 bouncing ball vectors

Start a new sketch. The variables **pos** (short for position) and **vel** (short for velocity) now can hold two bits of data. You can call them by using **.x** and **.y** after the name.

```
let pos
let vel

function setup()
{
  createCanvas(400, 400)
  pos = createVector(100, 100)
  vel = createVector(2.5, 4)
}

function draw()
{
  background(220)
  pos.add(vel)
  if ((pos.x > width) || (pos.x < 0))
  {
    vel.x *= -1
  }
  if ((pos.y > height) || (pos.y < 0))
  {
    vel.y *= -1
  }
  circle(pos.x, pos.y, 50)
}
```

Notes

This is the same as the previous except now we have the x and y velocity as a vector. This makes it much more elegant. In draw() the velocity component is added to the position.

Challenge

Change the initial start position and the relevant velocity components.

Sketch A10.3 the ball as an object

This is the same Sketch Aas the previous two examples, a circle bouncing off all the sides. We create a ball class which has a **constructor()** function, a **movement()** function and a **display()** function. You can give the functions other names, these are the ones I have chosen.

```
let ball

function setup()
{
  createCanvas(400, 400)
  ball = new Ball()
}

function draw()
{
  background(220)
  ball.movement()
  ball.display()
}

class Ball
{
  constructor()
  {
    this.pos = new createVector(100, 100)
    this.vel = new createVector(2.5, 5)
  }

  movement()
  {
    this.pos.add(this.vel)
```

```
    if ((this.pos.x > width) || (this.pos.x < 0))
    {
      this.vel.x *= -1
    }
    if ((this.pos.y > height) || (this.pos.y < 0))
    {
      this.vel.y *= -1
    }
  }

  display()
  {
    circle(this.pos.x, this.pos.y, 50)
  }
}
```

Notes

The ball is now a class 'Ball' with a constructor() function where you create the two vectors, the movement() and display() functions attribute the necessary characteristics to that ball.

Challenge

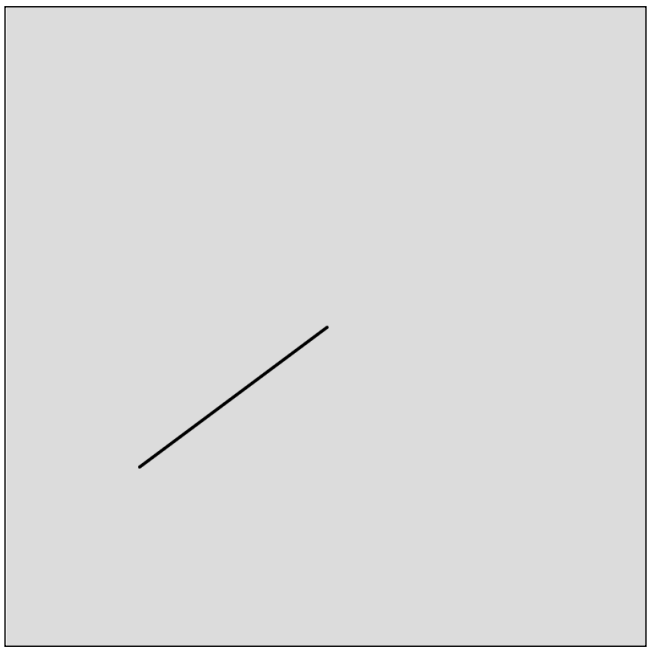
How would you create two balls bouncing around the canvas?

Sketch A10.4 the art of subtraction

Some vector maths using subtraction, subtracting two vectors. This example subtracts the centre of the canvas from the position of the cursor.

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

function draw()
{
  background(220)
  let pointer = createVector(mouseX, mouseY)
  let middle = createVector(width/2, height/2)
  pointer.sub(middle)
  translate(width/2, height/2)
  line(0, 0, pointer.x, pointer.y)
}
```

Vector subtraction



Notes

This subtracts two vectors. This code demonstrates the a simple application. Can you think of others?

Challenge

Comment out the line // pointer.sub(middle) and see what happens, this will demonstrate what that line of code does. Work out why

Sketch A10.5 vector multiplication

More vector maths with multiplication

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

function draw()
{
  background(220)
  let pointer = createVector(mouseX, mouseY)
  let middle = createVector(width/2, height/2)
  pointer.sub(middle)
  pointer.mult(1.5)
  translate(width/2, height/2)
  line(0, 0, pointer.x, pointer.y)
}
```

Notes

The .mult() function multiplies the two vectors

Challenge

Multiply by different amounts, e.g. 0.5, -1.5 etc

Sketch A10.6 magnitude of vector

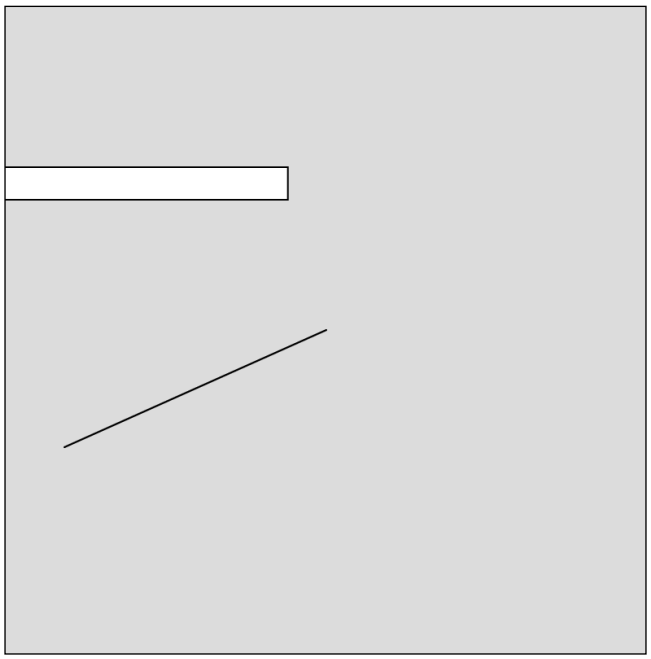
Remove:

pointer.mult(1.5)

You can return the magnitude of a vector

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

function draw()
{
  background(220)
  let pointer = createVector(mouseX, mouseY)
  let middle = createVector(width/2, height/2)
  pointer.sub(middle)
  let m = pointer.mag()
  rect(0, 100, m, 20)
  translate(width/2, height/2)
  line(0, 0, pointer.x, pointer.y)
}
```

Top bar measures the magnitude



Notes

The variable m is the magnitude of the line calculated by subtracting the mouse co-ordinates and the centre co-ordinates made so much simpler through creating vectors.

Challenge

Can you make a circle change colour with the magnitude of the line?

Sketch A10.7 feeling normal again

Remove:

```
let m = pointer.mag()
rect(0, 100, m, 20)
```

To normalise something is to make it a value of 1

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

function draw()
{
  background(220)
  let pointer = createVector(mouseX, mouseY)
  let middle = createVector(width/2, height/2)
  pointer.sub(middle)
  pointer.normalize()
  pointer.mult(150)
  translate(width/2, height/2)
  line(0, 0, pointer.x, pointer.y)
}
```

Notes

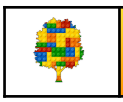
Using .normalize() normalises a value makes any vector value equal to 1.

Challenge

Comment out the line:

```
// pointer.mult(150)
```

You can now see how normalising makes it equal to 1, try different values for .mult()



Sketch A10.8 random position velocity

Creates an initial random value for its position and velocity

```
let ball

function setup()
{
  createCanvas(400, 400)
  ball = new Ball()
}

function draw()
{
  background(220)
  ball.move()
  ball.edges()
  ball.display()
}

class Ball
{
  constructor()
  {
    this.position = createVector(random(width), random(height))
    this.velocity = createVector(random(-5, 5), random(-5, 5))
  }

  move()
  {
    this.position.add(this.velocity)
  }
```

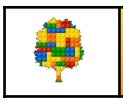
```
  display()
  {
    circle(this.position.x, this.position.y, 50)
  }

  edges()
  {
    if (this.position.x > width)
    {
      this.position.x = 0
    }
    else if (this.position.x < 0)
    {
      this.position.x = width
    }
    if (this.position.y > height)
    {
      this.position.y = 0
    }
    else if (this.position.y < 0)
    {
      this.position.y = height
    }
  }
}
```

Notes

Every time you run this Sketch Ait randomly choses a different x and y velocity component and starts from a random position. We have a constructor() function, a display(), move() and edges() function.

Challenge

Create an ellipse that has random dimensions so that every time you run the Sketch Ait is a different shape. hint: create a vector that has x dimensions and y dimensions



Sketch A10.9 setting a speed limit

You can limit the vector

```
let ball

function setup()
{
  createCanvas(400, 400)
  ball = new Ball()
}

function draw()
{
  background(220)
  ball.move()
  ball.edges()
  ball.display()
}

class Ball
{
  constructor()
  {
    this.position = createVector(width/2, height/2)
    this.velocity = createVector(0, 0)
    this.acceleration = createVector(0.05, -0.01)
    this.velocitylimit = 10
  }

  move()
  {
    this.velocity.add(this.acceleration)
```

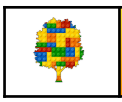
```
    this.velocity.limit(this.velocitylimit)
    this.position.add(this.velocity)
  }

  display()
  {
    circle(this.position.x, this.position.y, 50)
  }

  edges()
  {
    if (this.position.x > width)
    {
      this.position.x = 0
    }
    else if (this.position.x < 0)
    {
      this.position.x = width
    }
    if (this.position.y > height)
    {
      this.position.y = 0
    }
    else if (this.position.y < 0)
    {
      this.position.y = height
    }
  }
}
```

Notes

This adds an acceleration component to the velocity so it just gets faster and faster. The **.limit()** function limits the vector to a specified value, in this case it is the speed of the ball.

Challenge

Play around with the values of the velocity vector and wonder why it has that effect e.g. (0, -10), (10, 10) and so on



Sketch A10.10 a wandering I will go

This is a demonstration of how you can bring many elements to one object

```
let ball

function setup()
{
  createCanvas(400, 400)
  ball = new Ball()
}

function draw()
{
  background(220)
  ball.move()
  ball.edges()
  ball.display()
}

class Ball
{
  constructor()
  {
    this.position = createVector(width/2, height/2)
    this.velocity = createVector()
    this.acceleration = createVector()
    this.velocitylimit = 5
  }

  move()
  {
    let angle = random(6)
```

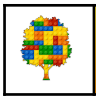
```
    this.acceleration = createVector(cos(angle), sin(angle))
    this.acceleration.mult(random(2))
    this.velocity.add(this.acceleration)
    this.velocity.limit(this.velocitylimit)
    this.position.add(this.velocity)
  }

  display()
  {
    circle(this.position.x, this.position.y, 50)
  }

  edges()
  {
    if (this.position.x > width)
    {
      this.position.x = 0
    }
    else if (this.position.x < 0)
    {
      this.position.x = width
    }
    if (this.position.y > height)
    {
      this.position.y = 0
    }
    else if (this.position.y < 0)
    {
      this.position.y = height
    }
  }
}
```

Notes

Has a pleasing almost alive feel about the movement. This shows how the vectors combined with classes starts to become a powerful tool.

Challenge

Just play around cutting and pasting or adjusting the values. This will give you a feel for the effects



Sketch A10.11 it's following you

How you can use classes to interact with a shape. Remove **edges()**

```
let ball

function setup()
{
  createCanvas(400, 400)
  ball = new Ball()
}

function draw()
{
  background(220)
  circle(mouseX, mouseY, 60)
  ball.move()
  ball.display()
}

class Ball
{
  constructor()
  {
    this.position = createVector(width/2, height/2)
    this.velocity = createVector()
    this.acceleration = createVector()
    this.velocitylimit = 5
  }

  move()
  {
    let pointer = createVector(mouseX, mouseY)
```

```
    this.acceleration = pointer.sub(this.position)
    this.acceleration.setMag(0.2)
    this.velocity.add(this.acceleration)
    this.velocity.limit(this.velocitylimit)
    this.position.add(this.velocity)
  }

  display()
  {
    circle(this.position.x, this.position.y, 50)
  }
}
```

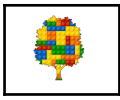
Notes

Move your mouse over the canvas and the ball will circle round you following you. You set the magnitude using the setMag() function.

Move the cursor to a point on the canvas!

Challenge

Change the velocity limit and the magnitude



Sketch A10.12 they're flocking to you

Let's have lots of them chasing you

```
let balls = []

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 20; i++)
  {
     balls[i] = new Ball()
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < balls.length; i++)
  {
    balls[i].move()
    balls[i].display()
  }
}

class Ball
{
  constructor()
  {
    this.position = createVector(random(width), random(height))
    this.velocity = createVector()
    this.acceleration = createVector()
    this.velocitylimit = 5
```

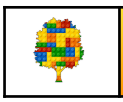
```
  }

  move()
  {
    let pointer = createVector(mouseX, mouseY)
    this.acceleration = pointer.sub(this.position)
    this.acceleration.setMag(0.2)
    this.velocity.add(this.acceleration)
    this.velocity.limit(this.velocitylimit)
    this.position.add(this.velocity)
  }

  display()
  {
    circle(this.position.x, this.position.y, 20)
  }
}
```

Notes

As above but this time using an array to have a selection of balls following you, all operating individually. It is quite fun and would provide some creative ideas.

Challenge

Increase the number of balls or add some random colour to them

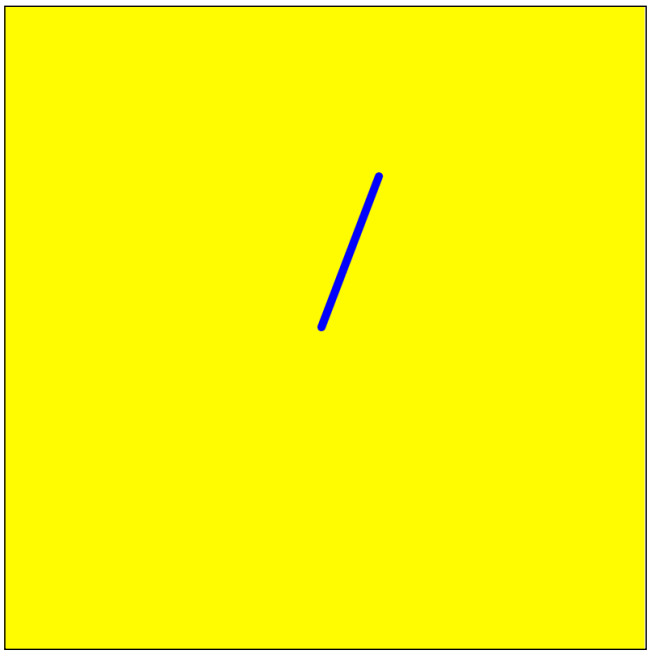


Sketch A10.13 random vector

We can create random vectors using p5.Vector.random2D(). It gives a value between 0 and 1, hence we multiply by 100

```
let arrow

function setup()
{
  createCanvas(400, 400)
  background('yellow')
  stroke('blue')
  strokeWeight(5)
  translate(width/2, height/2)
  arrow = p5.Vector.random2D()
  arrow.mult(100)
  line(0, 0, arrow.x, arrow.y)
}
```

A random vector is created every time you refresh



Putting vectors and classes together

1. we create an empty array of balls
2. we create 20 balls from the class Ball
3. we create a different (random) starting point for each ball
4. create vector for velocity in x and y direction
5. create a vector for acceleration for x and y
6. set a speed limit
7. draw and display the balls (circles)

```
let balls = []

function setup()
{
  createCanvas(400, 400)
  for (let i = 0; i < 20; i++)
  {
    balls[i] = new Ball()
  }
}

function draw()
{
  background(220)
  for (let i = 0; i < balls.length; i++)
  {
    balls[i].movement()
    balls[i].display()
  }
}

class Ball
{
  constructor()
```

```
  {
    this.pos = createVector(random(width), random(height))
    this.vel = createVector()
    this.acc = createVector()
    this.speedlimit = 6
  }

  movement()
  {
    let pointer = createVector(mouseX, mouseY)
    this.acc = pointer.sub(this.pos)
    this.acc.setMag(0.2)
    this.vel.add(this.acc)
    this.vel.limit(this.speedlimit)
    this.pos.add(this.vel)
  }

  display()
  {
    circle(this.pos.x, this.pos.y, 20)
  }
}
```

Notes

As above but this time using an array to have a selection of balls following you, all operating individually. It is quite fun and would provide some creative ideas.

Challenge

Increase the number of balls
Add some random colour to them
Increase the speedlimit value