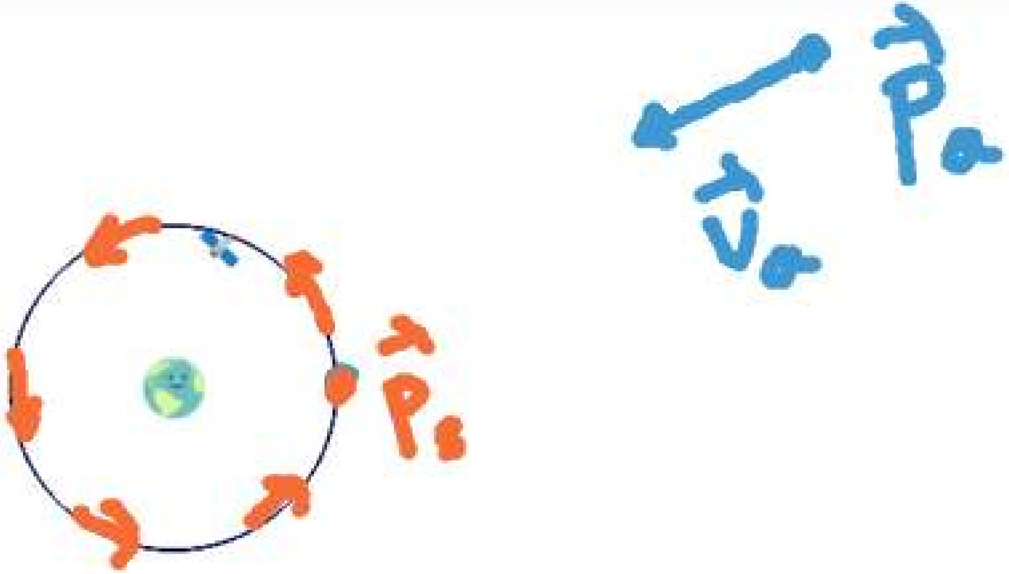
Mission: determine if there is a collision between asteroid and satellite.

$$\vec{p}^{k+1} = \vec{p}^{k} + \Delta t \; \vec{v}(\vec{p}^{k}, t^{k}).$$

- **satellite:** $\vec{v}_s(\vec{p}, t) = (-y, x)$ where $\vec{p} = (x, y)$, starts at $\vec{p}_s = (25, 0)$.
- **asteroid:** $\vec{v}_a(\vec{p}, t) = (-30, -2.5)$, starts at $\vec{p}_a = (50, 25)$,
- use $\Delta t = 0.2$.
- Collision if distance is less than 2 (in provided units).
- Should we fire thrusters to avoid asteroid? (fuel is expensive!)

t	ps	vs(ps,t)	pa	va(pa,t)	distance
0	(25, 0)	(-0, 25)	(50, 25)	(-30, -2.5)	35.4
0.2	(25, 5)	(-5, 25)	(44, 24.5)	(-30, -2.5)	27.2
0.4	(24, 10)	(-10, 24)	(38, 24)	(-30, -2.5)	19.8



Revisiting the Pixar ball from last lecture & lab.

- forces act on objects
- change in motion over some time is proportional to sum of forces acting on object.

$$\vec{a} = \frac{\Delta \vec{v}}{\Delta t}$$

input: initial position $\vec{p}_0 = \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}$

velocity $\vec{v}_0$

$$m\vec{a} = \sum \vec{f}$$

analytic solution

$$\boxed{y = y_0 + v_{y,0}\, t - \frac{1}{2} g t^2}$$

$$\vec{f}_g = m\vec{g}$$

$$\vec{g} = \begin{pmatrix} 0 \\ -9.81 \\ 0 \end{pmatrix}$$

-9.81 — g

$+y$

$y=0$

Things get more complicated with other forces.

add drag

$$\vec{F}_d = -\frac{1}{2}\rho A \|\vec{v}\|^2 C_d \frac{\vec{v}}{\|\vec{v}\|}$$

density of air

area

0.5 for a sphere

$$\vec{F}_g = m\vec{g}$$

$$\vec{g} = \begin{pmatrix} 0 \\ -9.81 \\ 0 \end{pmatrix}$$

analytic solution ???

+y

y=0

Let's just use a computer...

$$\vec{v}^k = \frac{\Delta p}{\Delta t} = \frac{\vec{p}^{k+1} - \vec{p}^k}{\Delta t}$$

$$\vec{a}^k = \frac{\Delta v}{\Delta t} = \frac{\vec{v}^{k+1} - \vec{v}^k}{\Delta t}$$

k is the iteration (frame) in animation

$$\vec{p}^{k+1} = \vec{p}^k + \Delta t \vec{v}^k$$

$$\vec{v}^{k+1} = \vec{v}^k + \Delta t \vec{a}^k$$

Newton's second law

$$m\vec{a} = \sum \vec{f}$$

$$\vec{a} = \frac{1}{m} \sum \vec{f}$$

Draw particles with `gl.drawArrays` and `gl.POINTS`.

JS

```
gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);
gl.vertexAttribPointer(a_Position, 3, gl.FLOAT, false, 0, 0);
gl.drawArrays(gl.POINTS, 0, nVertices);
```

new! new!

VS

```
attribute vec4 a_Position;

uniform mat4 u_ProjectionMatrix;
uniform mat4 u_ViewMatrix;

void main() {
  gl_Position = u_ProjectionMatrix * u_ViewMatrix * vec4(a_Position, 1);
  gl_PointSize = 50.0 / gl_Position.w;
}
```

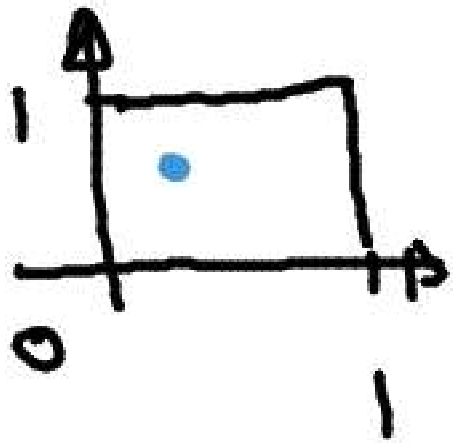
new!

FS

```
uniform sampler2D tex_Sprite;

void main() {
  gl_FragColor = texture2D(tex_Sprite, gl_PointCoord);
  //gl_FragColor = vec4(1, 1, 1, 1);
}
```

new!



We'll use *transform feedback* to capture updated position & velocity to a buffer and send to next iteration.

buffer for current position $\boxed{p^k}$ initialize with particle positions.

swap at the end of iteration.

buffer for updated position $\boxed{p^{k+1}}$ apply equations from Euler's method

$$\vec{p}^{\,k+1} = \vec{p}^{\,k} + \Delta t\,\vec{v}^{\,k}$$

in vertex shader capture this in a varying → use transform feedback.