U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Computer Graphics
MIEIC

Project – Auto Pilot

Circular animation

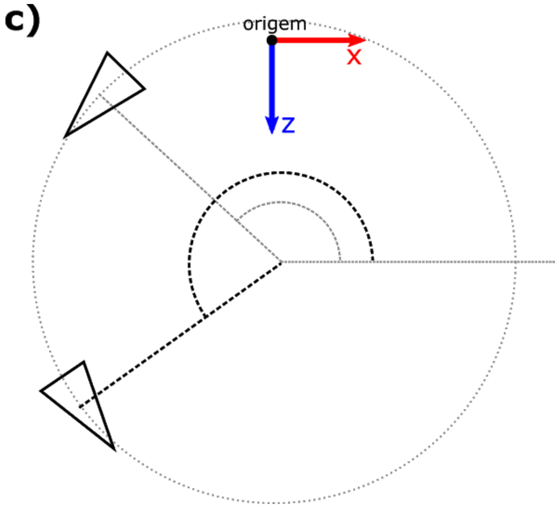
Teresa Matos
30 April – 1 May 2020

Auto pilot animation

At any frame, the “auto pilot” mode for the vehicle may be activated.

This mode represents a circular animation that:

- **Starts** in the current position and orientation
- Has **radius** of 5 units
- Each **lap** takes 5 seconds



Steps for auto pilot animation

The steps for this animation are similar to the vehicle’s regular animation:

1. Define **initial** state
2. **Update** current state
3. **Apply** state in vehicle’s **display**

1 Define Initial State - Variables

When the autopilot mode is activated, the **initial state** is initialized:

Initial state of auto pilot is:

- **Center** of circular animation
- **Initial angle**, relative to X axis
- **Position**
- **Orientation**

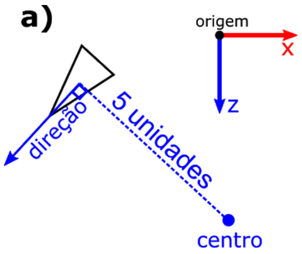
```
startAutoPilot(...){
… initialize other variables (e.g. radius)
   this.center = [Cx,Cy,Cz];
   this.pilotAngle = …;
   this.position = [Px,Py,Pz];
   this.orientation = …;
}
```

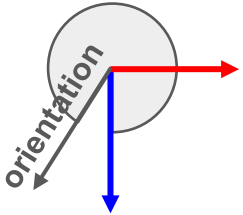
The position and orientation are **recalculated** using the animation's center and angle

1 Define Initial State - Center

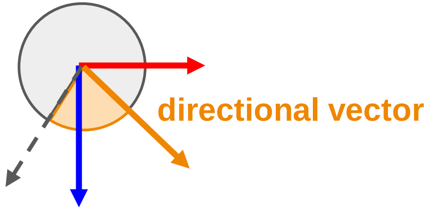
We obtain the **center** of the animation from the **initial position**, considering the **initial orientation**

- Calculate perpendicular orientation
- Calculate directional vector
- Apply directional vector **multiplied by radius**

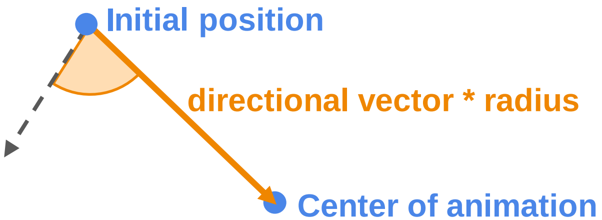




Initial orientation



Perpendicular orientation
(+ 90 degrees)

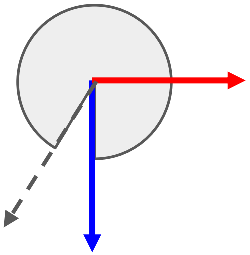


1 Define Initial State – Pilot Animation Angle

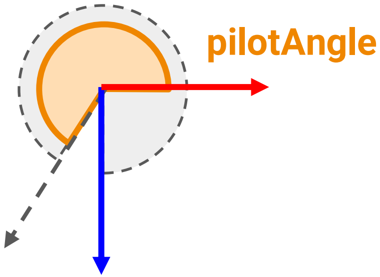
The angle around the center is initialized using the vehicle’s orientation

The **pilot angle** is relative to the **positive X axis**

```
this.pilotAngle = this.orientation - 90°
```



Current orientation,
relative to Z axis



Auto pilot’s initial angle

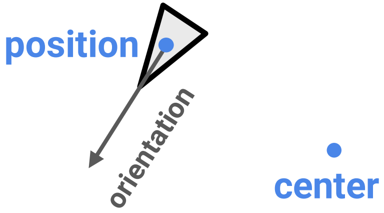
1 Define Initial State – Position

We now **recalculate** the position (same value, different methods)

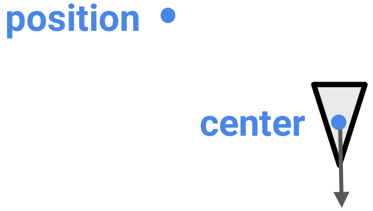
The position is obtained:

1. Starting at the center of the animation…

```
this.position = center
```



Unchanged position



Position is moved to **center**

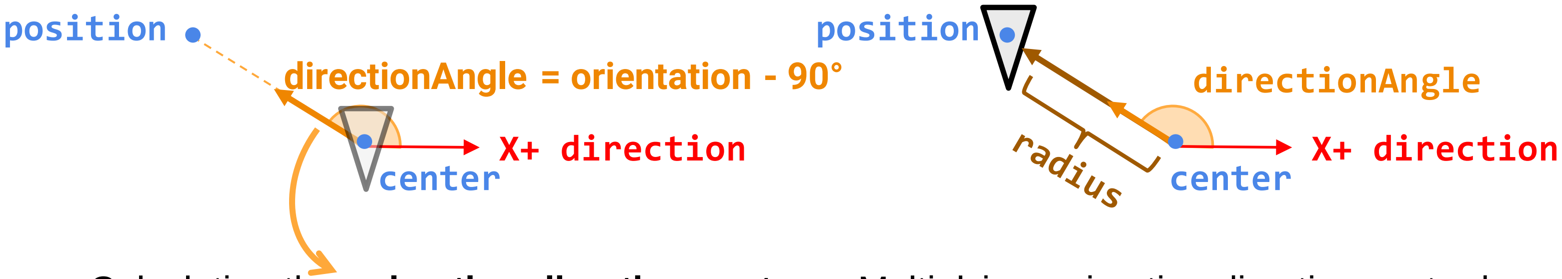
1 Define Initial State – Position

We now **recalculate** the position (same value, different methods)

The position is obtained:

2. … translating **radius** units in animation direction vector’s **direction**

```
this.position = center + directionVector * radius
```



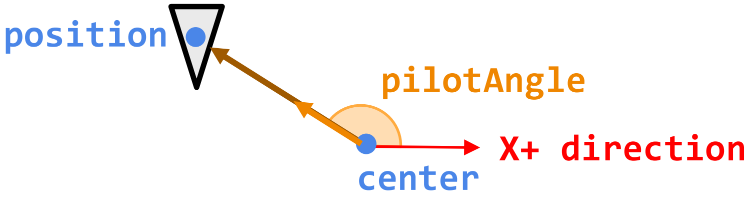
Calculating the **animation direction vector** Multiplying animation direction vector by **radius**

1 Define Initial State – Orientation

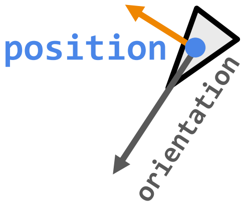
We now **recalculate** orientation (same values, different method)

The vehicle’s orientation is **perpendicular to pilotAngle**

```
this.orientation = pilotAngle + 90°
```



Recalculated position



Recalculated orientation

2 Update Current State – Speed

The position and orientation are calculated at every call for **update()**

We need to calculate the **angular speed**, considering the animation’s time

Considering that we want to perform a **complete rotation in 5 seconds**:

`angularSpeed = 360° / animationTime` (5 seconds)

2 Update Current State – Elapsed Time

To ensure that the **animation occurs in 5 seconds**, we need to update our state according to the elapsed time between frames

```
deltaTime = (currentTime - previousTime) / 1000;
```

To obtain deltaTime in seconds

The `update(t)` function from ***MyScene*** receives the current time (ms)

Time ***t*** is passed to the vehicle's `update(t)` function

2 Update Current State – Incremental Angle

Using the **elapsed time**, we calculate the angle to rotate between previous and current frame - **delta angle** (applying rule of 3 with angular speed)

```
deltaAngle = deltaTime * angularSpeed;
```

The **deltaAngle** is added to **pilotAngle**, which is used to **recalculate the position and orientation** (as done in the previous step).

3 Apply state in display

The `display()` function is not changed when applying the auto pilot mode

```
display(){
  translate(this.position);
  rotate(this.orientation);
  …
  objects.display();
}
```