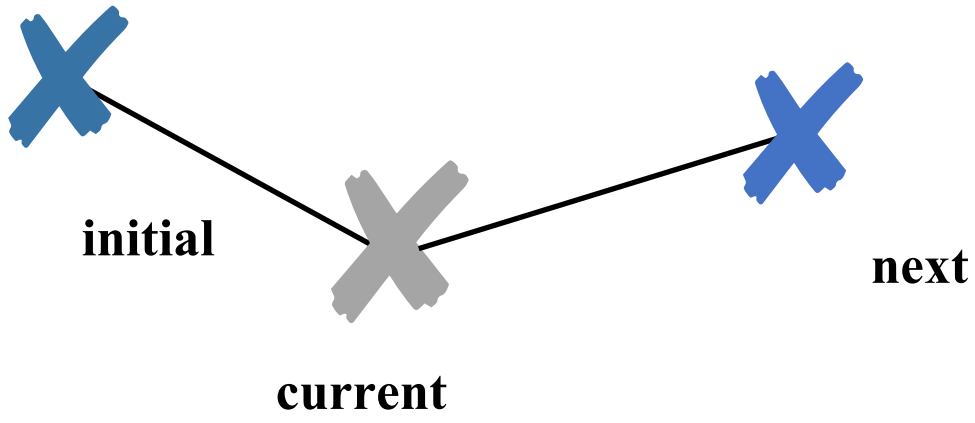
This app was designed to help blind students navigate around the Michigan State campus. The app uses data from an existing database to determine where sidewalks and intersections are located. There are some places in the database where the sidewalk markers were spaced far apart. In these cases, additional markers were created in between. It determines the location of a student on campus and maps a route to the building they select. It uses a quad tree algorithm to find the closest intersections to the sidewalk the student is on, to determine the path a student may take. The steps taken by the app are as follows:

1. Find intersections on path
    a. Get path: Line 162 initializes mRouteTask. Line 350 solves the route and places the result in mResults. Line 439 then gets the current route (curRoute) the user will take.
    b. Get intersections: Line 366 gets all intersections from the database. Line 372 parses the intersections and creates a list of intersections.
    c. Merge quad tree: Line 373 fills the quadtree with all of the intersections. Lines 385 through 403 build the quadtree of intersections. Lines 392-400 iterate through each intersection in the list. Line 393 creates a JSON object called geometry of an intersection. Line 394 gets the x coordinate of the intersection. Line 395 gets the y coordinate if the intersection. Line 397 adds the intersection to the quadtree.
    d. Merge intersections between path points: Line 459 densifies the path by adding points along the way to densify the path so all intersections are available.
2. Show turn at each intersection
    a. Get angle: Lines 556 through 599 determine the angle the user must take. Lines 571 and 574-585 determines the angle between the user's current path and the positive x axis. Lines 572 and 574-585 determines the angle between the path the user must take and the positive x axis. Lines 587 through 592 determine whether the user needs to take the acute or obtuse angle between the two paths.
    b. Convert to clock: Lines 603 through 607 convert the angle to hours on a clock. Math.ceil((finalAngle-15)/30) is the code used to scale the angle to an integer between 0 and 12. Lines 604 and 606 convert an angle of 0 to 12.

# Calculating Final Angle and Clock Direction Using 3 points

**Line 556:** goes through the list of intersection points
**Line 558:** go through the loop twice before indexing the list, to get three points

- initial at [i-2], the first point or coming from
- current at [i-1], second point where someone is
- next at [i], third point where someone is going




**Lines 566 & 568:** create two new points using the difference between the point for the coordinates

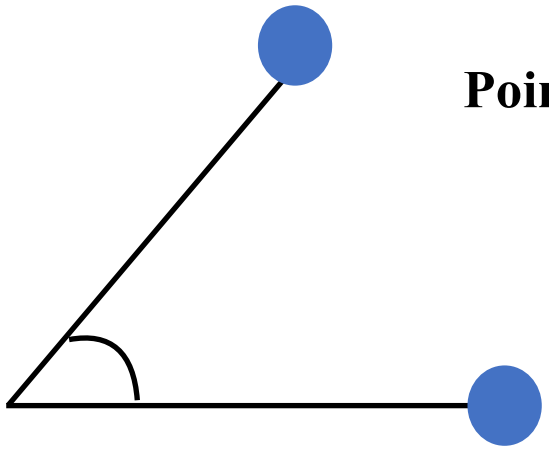
current.X - initial.X is the x coordinate of point 1
current.Y - initial.Y is the y coordinate of point 1

next.X - current.X is the x coordinate of point 2
next.Y - current.Y is the y coordinate of point 2

# Calculating Final Angle and Clock Direction Using 3 points

**Lines 571 & 572:** two angles are calculated. An alpha angle computed using point1 and a beta angle using point2.
The arctan2 function is used to compute the arc tangent of a point, then it is converted from radians to degrees using
(180/pi ) conversion. The absolute value is taken to ensure there are not any negative values.

**Example of**

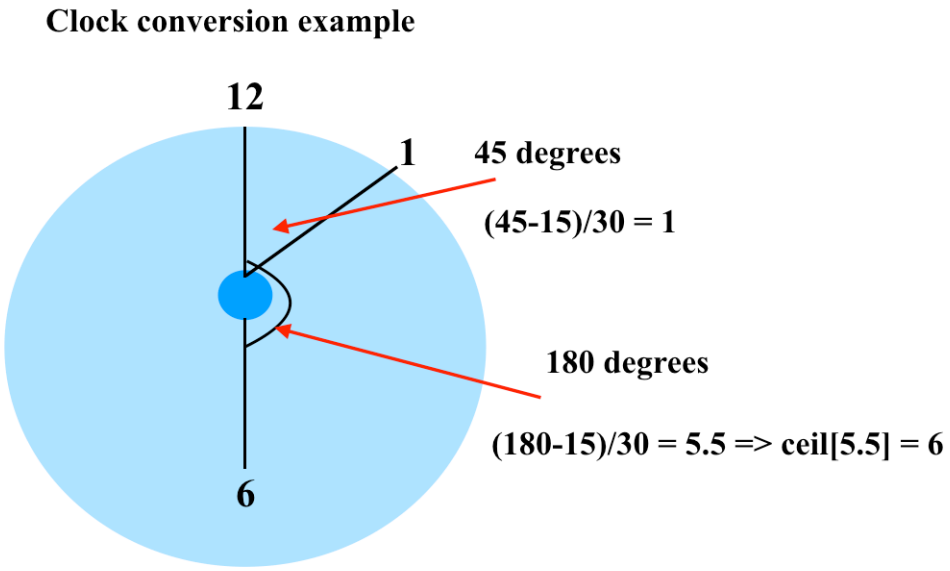



$$\operatorname{atan2}(y,x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{if } x > 0, \\ \frac{\pi}{2} - \arctan\left(\frac{x}{y}\right) & \text{if } y > 0, \\ -\frac{\pi}{2} - \arctan\left(\frac{x}{y}\right) & \text{if } y < 0, \\ \arctan\left(\frac{y}{x}\right) \pm \pi & \text{if } x < 0, \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0. \end{cases}$$

**Relevant math to Caitlin's code to get the finalAngle relative to direction**

# Calculating Final Angle and Clock Direction Using 3 points

**Line 603:** calculates the clock conversion using the finalAngle previously calculated. The direction the person is facing is considered to be 12 O'Clock. We do not convert for every clock time, we use ceiling to round up.




**Lines 604 - 607:** Checks if the finalAngle after the conversion is 0, if so make it 12 denoting that the person continues straight in the same direction they are currently facing.