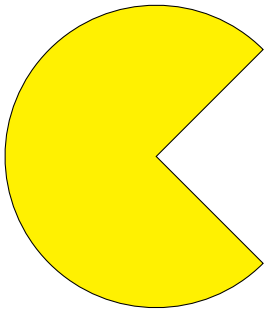
1. **Animating Pacman:**

   There is just one problem for section this week, but it is one of several parts. It also comes with a starter template, so make sure you have accessed the starter repository before getting going. All the below parts will focus on animating a simple scene of the classic Pacman character moving along a coridor, eating yellow pills. While this problem focuses on Pacman, several of the ideas used in this problem will apply directly to the Breakout Project, particularly for milestones 3 and 4.

   (a) If you run the `Problem.py` file in the starting repository, you will see that a simple scene has already been constructed for you, depicting a single corridor with eight yellow pills spaced down the corridor. You may notice that there is a blank space in the center, which is where you are to add a depiction of Pacman! Here you will start with Pacman facing to the left, where the mouth of Pacman indicates his facing direction and is comprised of a 90 degree missing wedge in a circle, as shown below.

   Add a single yellow, filled `GArc` to represent Pacman to the center of the window, which will place Pacman nicely in the gap between yellow pills.

   (b) Pacman moves, and so the next step is getting him moving! Add a `step` callback function which gets called by a timer event every 20 milliseconds. Each time `step` is called, it should simply move Pacman forward by the constant `PACMAN_SPEED`.

   (c) At the moment, Pacman will just move off the right side of the screen. While in the usual game, Pacman actually wraps around the screen when moving offscreen, here we will instead cause him to bounce off the wall. Doing so requires that we keep track of the direction Pacman is currently traveling. There are many ways to do this, but all of them will require updating that value within the `step` callback function (for instance, to indicate that Pacman should switch directions when his right side contacts the right side of the window). We know though that setting a variable within a function forces that variable to be a *local variable*, which would not persist between `step` calls. As such, this variable will need to be added to the `GWindow` (`gw`) as an *attribute*.

   Once you've decided what this attribute will be named, you have simply to build in the necessary logic to your `step` function. Doing so requires two things:

   - Updating how you are moving Pacman to incorporate the direction in some way. This might involve a comparison or multiplication, depending on how you are storing the direction of movement.

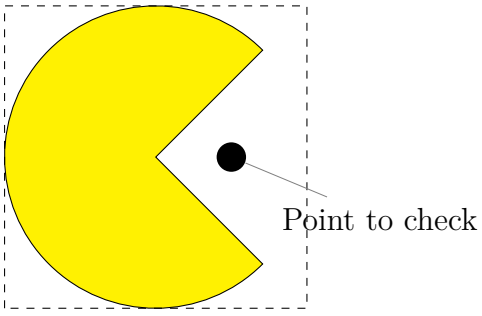
- Adding some logic that checks to see if Pacman has started to go off screen. This means checking to see if the right side of Pacman has started to go off the right side of the screen, or if the left side has started to go off the left side of the screen. The left side is a bit easier given how our coordinates work with `GArc`s, but since Pacman is currently facing to the right, you'll need to figure out both. When either condition occurs though, you want to do the same thing: reverse the direction that Pacman is moving.

Additionally, it looks a bit odd for Pacman to bounce off the right wall and then be moving seemingly backwards, with its mouth still facing to the right. You can fix this by updating the starting angle of your `GArc` using the `.set_starting_angle` and `.get_starting_angle` methods. What do you need to add to the starting angle of your arc to flip Pacman around? Implement this at the same time that Pacman's direction of travel changes, so that he seems to flip around when bouncing off a wall.

(d) Pacman is now moving, but he is just moving over the yellow pills. In practice, we want to make him seem to "eat" the pills by deleting the pills as Pacman moves over them. In a way, this is simple, as the `GWindow` has a method `.remove(obj)` that lets us remove any particular object. The problem here is that no variable has been assigned to each of the pills, and so we have no way of referring to which specific pill we want to remove.

In situations like this, it is frequently useful to take advantage of the `GWindow` method `.get_element_at(x,y)`. Here, you can provide an x and y coordinate in the window, and the `GWindow` will take care of looking at that location to see if there is any `GObject` currently located there. If there isn't, then the method will return `None`, but if there is, then the method will return the found object! If you assigned the output of `.get_element_at` to a variable, then you now have a way to refer to the object that you want to remove from the window!

Doing this requires checking a given location relative to Pacman each time `step` is called to see if a pill is present at that location. In our situation here, let's check the location about halfway into Pacman's mouth, as shown in the below image, where I have also included the bounding box for the arc:



Keep in mind that it may be useful to get the current x and y values of the Pacman arc when trying to work out the coordinates of this position. Then check to see if

there is a `GObject` at that location. If there is, then you should also check to ensure that the found object isn't Pacman himself or the background! But if it is neither, then it must be a pill, at which point you should remove it.

(e) **Challenge!** Currently, Pacman's mouth does not move as he moves, but in the arcade game he is constantly chomping. Add code to cause Pacman's mouth to seem to repeatedly open and close as he moves. Doing so will require adjusting the starting angle and sweep angle every time `step` is called. You'll need to change both to keep the chomping symmetric.