COMP270
Mathematics for 3D Worlds and Simulations

Week 5 Workshop Exercises: Horizontal Acceleration and Collisions

INTRODUCTION

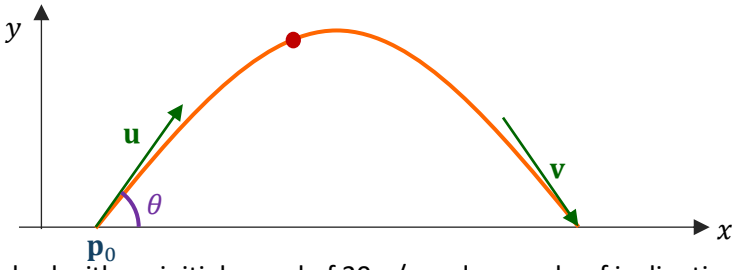
This worksheet is split into two sections: Part A is a set of "traditional" maths questions, the answers to which will be demonstrated in the workshop (though feel free to try them yourself first; you may find it helpful to refer to last week's exercises). Part B is an exploration of collision detection algorithms.

PART A

Answer the following questions using pen(cil) and (graph) paper.
Pro tip: show your working – diagrams can be helpful!

For the following exercises, unless otherwise stated, assume that the acceleration due to gravity is 9.81m/s^2 acting in the negative y direction (with the y-axis pointing upwards), and there is no air resistance or other force acting on the objects except where stated.

Diagram of a general projectile motion set-up



1. A projectile is launched with an initial speed of 30m/s and an angle of inclination θ = 40° from the initial position $\mathbf{p_0} = (0\text{m}, 2.5\text{m})$, i.e. 2.5m above the origin, which is at ground level. There is a constant horizontal acceleration due to wind of -7.5ms^{-2} acting on the projectile.
    a. What is the initial velocity $\mathbf{u}$ in vector form?
    b. At what time will the projectile reach its apex (highest point)?
    c. What are the coordinates of the projectile at the apex?
    d. How long will it take for the projectile to come back to an altitude of $y = 2.5\text{m}$?
    e. What will the horizontal displacement be at this time?
    f. For how long is the projectile in the air before it hits the ground (y=0)?
    g. Find the values for the projectile's
        i. final velocity, $\mathbf{v}$, and
        ii. final horizontal displacement, s'_x

    when it hits the ground. Is this the farthest distance along the horizontal that the particle reaches?

COMP270
Mathematics for 3D Worlds and Simulations

Week 5 Workshop Exercises: Horizontal Acceleration and Collisions

PART B

The Bitbucket repository below contains the code for the beginnings of a basic Asteroids game, which involves manoeuvring a "spaceship" around the screen and firing bullets at space rocks to break them into smaller fragments:

https://gamesgit.falmouth.ac.uk/projects/COMP270/repos/comp270-collisions-workshop

The following features have already been implemented:

- Movement of the player's ship with the following controls:
  - up arrow: applies a thrust in the direction the spaceship is facing
  - right/left arrow: rotates the spaceship clockwise/anticlockwise
  - space bar: fires a bullet from the front of the spaceship
- Generation and animation of the asteroids (with a constant velocity and rotation)
- Replacing an asteroid with smaller asteroids in a similar location, to represent fracturing due to bullet impact.

However, some core functionality is missing: amongst other things, there is no collision detection to say whether a bullet is intersecting an asteroid!

The function `Asteroid::pointIsInside()` should return true if the input `Point2D`, `point`, is enclosed by the edges of the `Asteroid` shape, as defined by the positions of its vertices in world space, stored in the `m_worldVerts` array.

1. The shape of an asteroid is relatively complex, but we've seen that it's possible to approximate complex shapes with simpler ones. Implement algorithms to find the dimensions of the following bounding shapes, and cause `Asteroid::pointIsInside()` to return true if the bullet at position `point` is inside the bounding shape:
   a. A bounding **box** (i.e. a rectangle whose minimum vertex matches the asteroid's minimum vertex coordinates and whose maximum matches the asteroid's maximum vertex coordinates)
   b. A bounding **circle** (i.e. a circle centred at the average of the asteroid's vertex positions, with a radius just large enough to enclose all the asteroid's vertices).

   Which of the two bounding shapes do you think is most appropriate for the asteroids?

2. You may notice that sometimes asteroids are breaking if a bullet goes close to them but doesn't actually hit, in which case a more accurate intersection test is required.
   a. You can read about a variety of intersection tests here: https://erich.realtimerendering.com/ptinpoly/
   Choose one (or more) appropriate for the asteroid and implement it/them in `Asteroid::pointIsInside()`

b. How does the accuracy and speed of the test you've just implemented compare with the bounding shape tests?
   Try running the more accurate test with and without using the bounding shape test first, to eliminate any clear misses; do you notice a difference in the speed at which the game runs?