COMP270
Mathematics for 3D Worlds and Simulations

Week 5 Seminar: Algebra and Collisions (Rearranging things!)

INTRODUCTION

This week's exercises are in two varieties: some recommended questions to practice algebraic rearrangement of equations, and some topics to discuss based on this week's lecture content.

ALGEBRA

Hint: you can find the standard trigonometric identities, which you are free to use in your proofs, here.
You might also like to read the following guides for working with algebraic expressions:

- https://www.mathplanet.com/education/algebra-2/equations-and-inequalities/solve-equations-and-simplify-expressions
- https://www.chegg.com/study-101/how-to-study-math-algebra/
- https://sciencing.com/tips-for-solving-algebraic-equations-13712207.html

You may find questions 3b and 4b especially useful for worksheet 2…

1. Prove that $(2n+3)^2 - (2n-3)^2$ is a multiple of 8, for all positive integer values of n.
2. Prove algebraically that:
   a. The difference between the squares of any two consecutive integers is equal to the sum of these two integers.
   b. The sum of the squares of any two consecutive numbers always leaves a remainder of 1 when divided by 4.
3. Solve the following pairs of simultaneous equations:
   a. $x^2 + 3y^2 = 31$ and $2x^2 + 5y^2 = 53$
   b. $3x^2 + 2xy = 11$ and $5xy - 2x^2 + 2y = 26$
4. Prove the following trigonometric identities:
   a. $\tan^2\theta - \sin^2\theta \equiv \tan^2\theta\sin^2\theta$
   b. $\frac{1+\cos\theta+\cos 2\theta}{\sin\theta+\sin 2\theta} \equiv \cot\theta$ (with $\cot\theta = \frac{1}{\tan\theta}$)
   c. $4\sin\theta\cos\theta \equiv \frac{\sin 4\theta}{\cos 2\theta}$
5. Solve the trigonometric equation given by $\cos 2\theta\cos\theta - \sin 2\theta\sin\theta = 0$ for $0 \le \theta \le 2\pi$

NB you can find plenty more practice exercises like these online, including at the following links:
https://www.analyzemath.com/high_school_math/grade_12/trigonometry.html
https://www.mathsgenie.co.uk/resources/111_proof.pdf

COMP270
Mathematics for 3D Worlds and Simulations

Week 5 Seminar: Algebra and Collisions (Rearranging things!)

COLLISIONS

The questions below are based on the Asteroids game from the workshop, which you can clone from the Bitbucket repository:

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

Don’t worry if you haven’t completed the workshop exercises, as you won’t need to do any coding in this session, but you might like to download and run the basic game before completing the activities.

The workshop exercises involved implementing a basic (or even not-so-basic) algorithm to check for collisions between a bullet (represented as a point) and an arbitrary polygonal shape, including by approximating the shape with a simpler circle or box, which is a fundamental technique in speeding up collision processing.

There are several further optimisations and enhancements that could be made to the game, some of which are listed below. In your groups, choose one or more features (from the list or of your own invention) and consider how you might go about implementing them, thinking about the computational steps involved along with any formulae, physical laws, algorithms or data structures you might need to use.

- Your ship has been upgraded! Now you can fire lasers instead of bullets; how would you adapt the collision detection to work with a continuous energy stream instead of a single projectile?

- At present, only collisions between bullets and asteroids are implemented, which means that asteroids may pass through other asteroids and the player’s ship.

    - Outline a technique for detecting collisions between the following combinations of object, considering whether or not the same approach should be used for both:
        - Two asteroids
        - An asteroid and the player’s ship

    - What kind of response would you expect for each of the colliding object combinations above? Describe the physical behaviours the objects should display and what computations would be involved in creating them.

- Besides using bounding shapes for initial collision detection, how could you speed up testing for collisions? (For example, to avoid checking every object against every other).