# CSC3222 Coursework

Welcome to CSC3222! This module is aimed at providing you with an understanding of video game simulation techniques. The areas investigated in this module can broadly be classified as being either those that enhance the physical accuracy of your game simulation, or those that enhance the appearance of intelligence and decision making in the agents within the simulation. The lecture series will introduce and cover the theory behind these concepts, while the tutorial series will look at the practical implications of implementing these concepts, specifically in the context of the coursework specification. Together, these should provide you with experience with the module concepts, and the skills required to apply them successfully.

**Module Assessment**

There are 2 pieces of assessment for this module:

* + Coursework project 1, submitted via NESS (30)
  + Coursework project 2, submitted via NESS (70)

The coursework projects are designed to be complementary to each other, and are to be completed by modifying and extending a starter codebase which implements a basic version of the game design specification outlined below, without any of the parts relating to the module concepts implemented. Coursework project 1 will focus on the physics aspects of gaming simulations (correct motion throughout an environment, and correct detection of collisions that occur between objects), while coursework project 2 covers the correct reaction to collisions, navigating through an environment, and making decisions based on the state of the environment. While both pieces of coursework share a codebase, and will be implemented as part of the same overall Visual Studio solution, you must make two NESS submissions – once for the coursework 1 submission, and once for coursework 2.

# Project 1 (Deadline: 16:00, Friday 12th March 2020)

# Aim

The purpose of this coursework is to implement the physics and collision detection calculations into the simulation to correctly meet the requirements of the software specification. This will include the addition of Newtonian dynamics to the various game entities within the scenario using numerical integration, and the correct determination of collisions within the simulation, with the appropriate response.

#### Newtonian Dynamics

All moving entities should move via the application of Newtonian dynamics, manipulating the position and associated derivatives as necessary, taking into consideration the mass of the entity.

#### Collision Detection

The simulation should include the ability to detect collisions between spheres and axis aligned bounding boxes – which object should use which collision volume is outlined in the game specification. Detected collisions should initially be output to the console window.

# Marking Scheme

* 14 marks are available for correct implementation of Newtonian physics (application of forces, integration of position over time, springs and impulses on appropriate game entities)
* 16 marks are available for implementation of collision detection (sphere / sphere, sphere / AABB, AABB / AABB, broadphase and narrow phase)

**Deliverables**

* Zipped Microsoft Visual Studio Project, Source Code, and Executable (submitted via NESS).

**Learning Outcomes**

* Understand how to represent mathematical concepts in code
* Understand the implementation of Newtonian dynamics in a real-time simulation
* Deduce and apply the appropriate simulation technique given a particular simulation requirement.

# Project 2 (Deadline: 16:00, Thursday 21st May 2020)

# Aim

The purpose of this coursework is to add in the ability for collisions to be resolved with the correct physical responses, and to implement AI and pathfinding into the simulation to correctly meet the requirements of the software specification. The correct handling of game entities over the course of the simulation is also to be completed, to ensure that the efficiency and correctness of the simulation as a whole is maintained – entities should exhibit the behaviour outlined in the specification consistently, and efficiently.

#### Collision Response

Colliding entities should bounce off each other via the calculation of an impulse, unless the collision produces a gameplay effect (Enemies can kill the player, for instance).

#### AI

The enemy types outlined in the game specification should have the correct state-based behaviour

#### Pathfinding

One of the game objects is specified to use pathfinding to find the player, which should be implemented using an appropriate graph-searching algorithm.

# Marking Scheme

* 16 marks are available for implementation of collision resolution (impulse calculations, correct gameplay interactions in response to entity collisions)
* 20 marks are available for correct implantations of artificial intelligence (handling of entity state and decisions, use of flocking where appropriate)
* 24 marks are available for the addition of pathfinding (implementation of A\*, correct path calculations in response to changing conditions, efficient data structure usage)
* 10 marks are available for simulation consistency (handling of entity lifetime, application of gameplay rules)

**Deliverables**

* Zipped Microsoft Visual Studio Project, Source Code, and Executable (submitted via NESS).

**Learning Outcomes**

* Develop familiarity with the nature of embedded map data, and its role in influencing path planning
* Understand the fundamental algorithms behind effective path finding in games
* Demonstrate an understanding of the interplay between game logic and physics computation
* Demonstrate correct handling of resources over time in a real-time simulation

# Game Specification – Fruit Wizard!

You have been tasked with finishing off a 2D arcade game, in which the player controls a character using the arrow keys, and must collect randomly spawning food items around the play area, and bring them back to the shop to earn points. Preventing the player from doing this are a series of enemy robots that patrol the play area and will try and shoot the player. The player starts with 3 lives, and when they lose all of their lives the game is lost.

The game is partially implemented, but does not contain any of the AI, pathfinding, or physics elements, and it is up to you to implement them as you see fit – you can choose the speed at which the described entities move, how heavy the entities are, and how often their AI makes decisions.

# Game Entities

The game is comprised of a game map that fills the screen, made up of a series of platforms connected with ladders, and bounded by walls. The following game entities can also be added to the game world:

## The Player: Wizzy



The player controls their chosen player character using the arrow keys, and can walk around on platforms, and climb up ladders. Walls stop the player’s progress. The player can fall off platforms, and is affected by gravity, but suffers no ill effects from falling any height. The player starts with 3 lives. If the player is hit by an enemy, they lose a life, and must reset back to the start point in the bottom left of the map. If all lives are lost, the game resets, and the player begins again with a score of 0, and no Pixie Dust collected.

## Magic Spell



Being a wizard means that the player can cast spells! If the player presses space, they may cast a spell. This will launch in a random direction, where they will move at a constant velocity. If they hit the floor or a wall, they will bounce off it. After 3 of these bounces, the Spell should disappear. It should also disappear if its speed ever reaches 0. Spells collide with enemies, but no other entities. Spells are not affected by gravity. The player starts with 4 Spells.

## Enemy Type A: Guards



The game area contains 4 Guards, who will be distributed randomly throughout the level. If the player loses a life, the guards are randomly redistributed. Guards will wander from left to right – if they move more than 10 tiles from their starting point, or reach the edge of their platform, they will turn around. If the player is within 3 tiles of the guard, the Guard will move towards the player and try and hit the player with their spear. If the player moves away, or the guard has moved more than 10 tiles away from their starting position, they will return to their initial wandering.

If the Guard is hit by a Spell, the player will earn 200 points, and the Guard will be knocked backwards and be stunned for 5 seconds. If this causes them to fall off a platform, the tile they land on becomes their new starting point. While stunned a Guard cannot attack the player, and the player does not collide with them. Guards are lazy, and will not climb ladders.

## Enemy Type B: King Froggo



King Froggo will spawn at the top right of the map once the player has collected 16 pieces of fruit. He is *very* annoyed that the player is taking his fruit, and will chase the player across the map. King Froggo will use ladders to try and reach the player. He can also jump horizontally from platform to platform if necessary – he is quite lazy, though, and prefers not to. If the player is within 2 tiles he will periodically try and hit the player with his trident. If Pixie Dust spawns while King Froggo is on the map, he will try and protect it by moving towards it and standing on it, but will still try and attack the player if nearby. If King Froggo is hit by a Spell he is knocked backwards and is stunned for 3 seconds.

## Collectable Fruit



The game begins with 16 items of fruit placed randomly on platforms throughout the level. If the player touches a piece of fruit, it disappears, and the player earns 1000 points. Enemies do not interact with the fruit objects. If the player loses a life, the remaining fruit in the level are randomly redistributed. If the player collects all of the fruit, they earn an extra 5000 points, and all 16 pieces of fruit reappear, and are randomly distributed throughout the level. A new Guard will also be placed randomly in the map.

## Pixie Dust



Every 30 seconds, there is a chance that some Pixie Dust will spawn randomly in the map. Enemies do not collide with Pixie Dust. The player may collect the pixie dust to earn 500 points. If the player collects 4 pieces of Pixie Dust, they may press ctrl to activate Pixie mode, outlined below. If the player has 4 pieces of Pixie Dust, no more will spawn, until the player activates Pixie mode, or the player dies.

## Pixies



Activating Pixie mode will spawn 8 Pixies randomly throughout the level. They do not collide with the game world or enemies, but do collide with the player, where they can be collected to earn 1000 points, and increase the player’s Spell count by 1. The Pixies should move around the level using flocking rules to avoid any enemies on screen, and should also slowly move towards the player. If the player dies while Pixie mode is active all of the remaining Pixies will disappear.

## Ladder



Wizzy and King Froggo can use ladders to move from platform to platform. Both should move up and down the ladder at half the speed that they do on the platforms.