RMIT UNIVERSITY

# Games and Artificial Intelligence Techniques – COSC3145

## Assignment 1

| | |
|---|---|
| Assessment Type | Individual assignment.<br>Submit online via Canvas |
| Marks | 10 |

## Overview

The goal of this assignment is to give you practical, hands-on experience implementing steering behaviours and finite state machines (FSMs), two foundational AI techniques in game development.

You will work with a simplified 2D predator-prey simulation built in Python using Pygame, where:

- The player controls a frog that moves by clicking and can shoot bubbles.
- Flies swarm around in flocks and flee when threatened.
- Snakes patrol the arena and chase the frog when provoked.

The starter project already includes all artwork, rendering, and collision setup, but the AI logic for movement, decisions, and reactions is missing.
Your job is to implement and extend that logic to make the world feel alive and believable.

## Learning Outcomes

This assessment supports the following course learning objectives:

- **[CLO1]:** Apply AI algorithms and steering techniques to simulate agent behaviour.
- **[CLO2]:** Design and develop interactive agent systems using finite state machines and procedural movement.

By the end of this project, you will:

- Understand how steering forces create natural motion.
- Implement FSMs to manage dynamic agent states.
- Extend simple behaviours into reactive, predictive AI systems.

## Assessment Details

The assignment is divided into three main parts, 5 marks each !

### Part 1 - Steering Behaviours (3 marks)

You will implement core motion logic in the steering.py file and observe it applied across the frog, flies, and snakes.

#### A- Implementing Arrive

In the starter game, the frog moves toward a clicked target position.
Currently, it overshoots the destination because its motion is linear.
You must implement smooth arrival behaviour that:

- Slows the frog down as it gets close to the target ("slow radius").
- Stops the frog completely within a small "stop radius."
- Produces smooth, frame-rate-independent motion (use delta time).

This will make the frog's motion appear natural and responsive rather than abrupt.

#### B- Boids Flocking

A group of flies has been set up in the world. Each fly:

- Calculates steering forces for separation, cohesion, and alignment with nearby flies.
- Blends these forces to maintain a cohesive group.

In the starter code, these components currently return neutral forces, meaning the flies simply drift randomly.
You must implement:

- **Separation:** Flies steer away from nearby neighbors to avoid crowding.
- **Cohesion:** Flies move toward the average position of nearby flies to keep the group together.
- **Alignment:** Flies match their velocity direction with their neighbors.

When implemented correctly, flies will form an organic flock, breaking apart when threatened and regrouping afterward.

#### C- Fleeing Behaviour

Flies must also react when the frog (or its bubbles) comes close.
You will implement a flee force that:

- Steers directly away from a threat.
- Increases intensity when the threat is near.
- Combines with other flocking forces so flies scatter smoothly instead of jittering.

This connects to their FSM transitions, allowing flies to switch between calm flocking and panic flight.

#### D- Obstacle Avoidance

Snakes chase the frog using the seek behaviour, but without obstacle avoidance, they'll collide with trees.
You need to implement a seek-with-avoidance method that:

- Casts a circle "ray" from the snake to its target to check for obstacles.
- If the direct path is blocked, tests offset angles (left and right) to find a clear route.
- Returns a new steering vector that guides the snake around obstacles.

This is similar to the "corridor search" approach explained in Week 2, using incremental rotations until a safe path is found. When implemented, snakes will intelligently weave through the arena instead of stopping or colliding.

## Part 2 - Finite State Machines (FSMs) (2 marks)

Finite State Machines control how agents react to conditions and change behaviour dynamically. You will implement them in fly.py and snake.py. Below is the finite state machine (FSM) to be implemented for the snakes.

### Overview

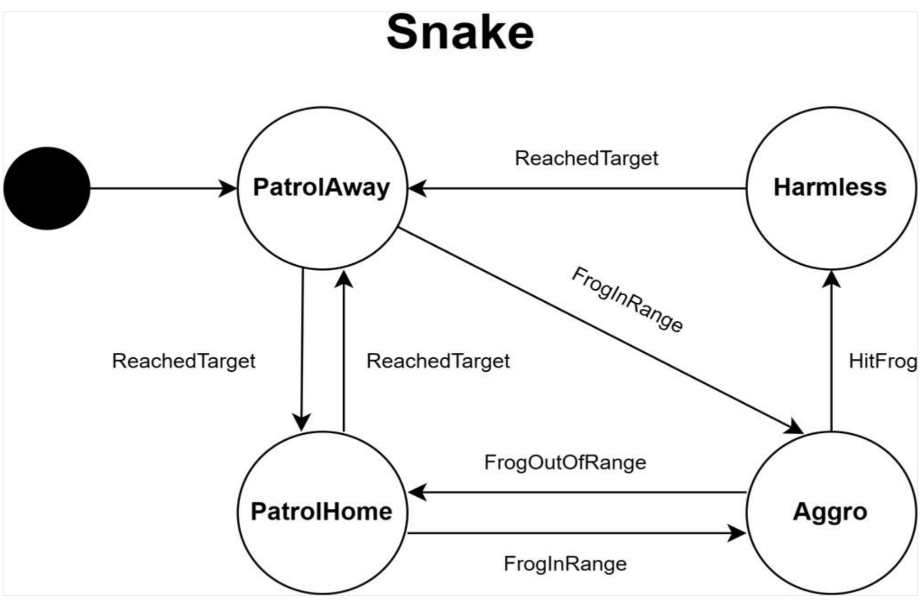


Figure 1- The finite state machine (FSM) to be implemented for the snakes.

### A- FLY FSM

Flies have two primary states defined in the code:

- **Flock:** Normal group behaviour, driven by boid forces.
- **Fleeing:** Triggered when the frog or bubbles are within a danger radius.

In the starter code:

- The transitions between these states exist but have no logic.
- The timer that controls how long flies stay scared is also empty.

You must implement:

1. Logic that checks distances to the frog and bubbles.
2. Smooth transitions from Flock → Fleeing → Flock again when safe.
3. Optional timer-based easing to prevent flickering between states

### B- Snake FSM

Snakes are designed with four states:

- **PatrolAway:** Moves from home toward a predefined patrol point using arrive.
- **PatrolHome:** Returns home when done patrolling.
- **Aggro:** Chases the frog using seek or pursuit.
- **Harmless:** After being hit by a bubble, stops chasing and returns home calmly.
-

In the starter code, these states exist but transitions are not active.
You must:

- Implement range checks (AggroRange, DeAggroRange) for when to switch between patrol and chase.
- Return to home when the chase ends or when pacified.
- Use arrive/seek logic to move between waypoints and targets.

This FSM defines the core predator behaviour in your game.

## Part 3 - Advanced AI Extensions (5 marks)

Once the base behaviours are working, implement these new, creative features that were not in the tutorial.
They are scaffolded in the code with TODO comments and placeholders.

## A- Pursue (1 mark)

Enhance snake intelligence by adding a **pursue** behaviour.
Instead of steering toward the frog's current position, predict where the frog will be next:

- Use the frog's velocity to estimate its future position.
- Steer toward that predicted point for more realistic pursuit.
- Replace the normal seek in the Aggro state with pursue.

This makes snakes capable of cutting off the frog rather than lagging behind.

## B- Evade (1 mark)

Implement an **evade** behaviour for flies.
This is the inverse of pursue:

- Predict where the frog will be based on its velocity.
- Flee from that predicted position instead of its current one.

Flies will now start dodging earlier and react more smoothly to fast-moving frogs.

## C- Confused State (1 mark)

Add a **Confused** state to the snake's FSM.
When a snake is hit by a bubble:

- Transition it from Aggro → Harmless → Confused.
- During Confused, the snake wanders randomly for a few seconds before returning home.
- Add a timer so the confusion period ends automatically.

This adds personality and realism to the snake's behaviour.

## D- Idle State (1 mark)

Add an **Idle** state to the fly's FSM.
When flies are far from the frog for a certain duration:

- Stop active flocking and switch to idle drift.
- Use a small random "wander" movement to simulate gentle hovering.
- Return to Flock state once the frog re-enters range.

This gives the ecosystem calm, believable downtime between interactions.

## E- Hurt State (1 mark)

Add a **Hurt** condition to the frog.
When the frog collides with an aggressive snake:

- Deduct one health point.
- Flash or tint the frog to indicate temporary invulnerability.
- Prevent further hits for ~1 second.

This makes gameplay fairer and gives visual feedback to the player

Additional note

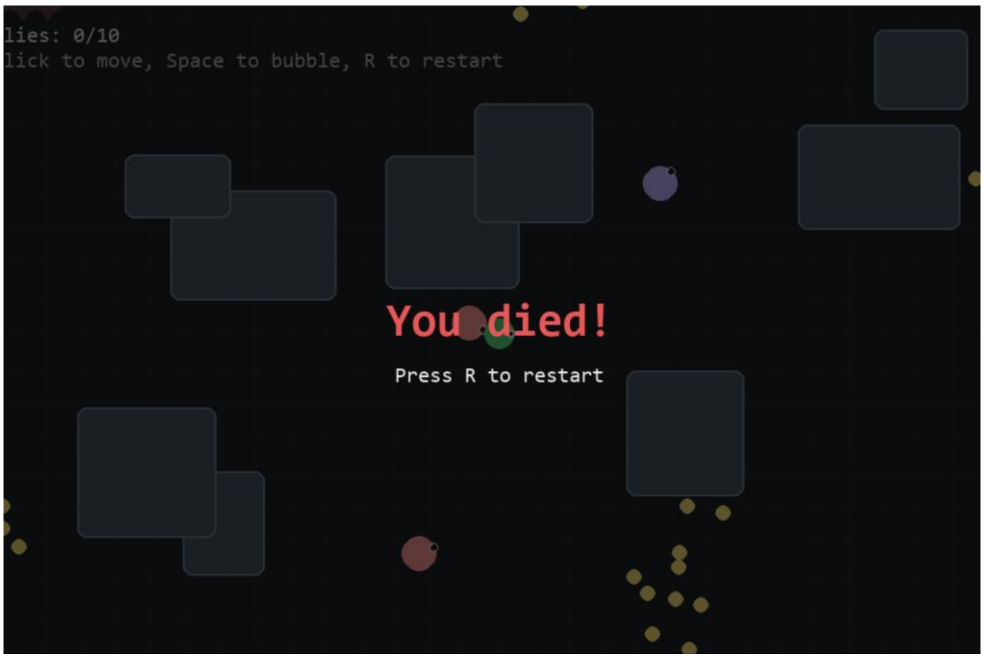
- **PatrolAway**: In this state, the snake should move towards a pre-defined "patrol point", using the *arrive* steering behaviour.
- **PatrolHome**: Same behavior as in PatrolAway, except that the snake should move towards its "home" position.
- **Aggro**: In this state, the snake should chase the frog via a seek steering behaviour.
- **Harmless**: In this state, the snake should move towards its "home" position via the *arrive* steering behaviour.

**Handling Health Bar and Fly counter**

- The player should start with 3 health and lose 1 health each time the frog collides with an aggroed snake.
- The fly counter should start at zero and increment each time the player catches a fly.

**Handling Game Over** (similar to the tutorials)

- Add a game over screen that indicates whether the player has won or lost and allows the game to be restarted.
- If the player runs out of health, the screen should read "You died!"
- If the player's fly count reaches 10, the game should end with the message "You won!".
- The game should pause on the game over screen, and the player should be able to restart the game by pressing 'R'.



```
lies: 0/10
lick to move, Space to bubble, R to restart
```

You died!

Press R to restart

## Submission

**What to submit on canvas ?**

1) A zip file containing your code
2) A video of you demonstrating the execution such as:
   Part 1: Task A, B, C, D
   Part 2: Task A, B
   Part 3: Task A, B, C, D, E

- Work only in the provided **A1_Starter** folder.
- Implement the missing logic in the given files inside the **A1_Starter**
- Late submissions will incur a penalty of 10% of the total score possible per calendar day.
- If you do submit late, please let us know so that we download the correct version of your assignment.
- Information on applying for special consideration is available here: https://www.rmit.edu.au/students/my-course/assessment-results/special-consideration- extensions/special-consideration

### Academic integrity and plagiarism (standard warning)

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarized, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods.
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites. If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviours, including:

- Failure to properly document a source.
- Copyright material from the internet or databases.
- Collusion between students.

For further information on our policies and procedures, please refer to the following: https://www.rmit.edu.au/students/student-essentials/rights-and-responsibilities/academic-integrity.

We will run code similarity checks.

### Marking Guide

The rubric used for marking will be provided on Canvas at the bottom of the assignment page.