Intelligent Systems Engineering
ECE 449
Fall 2025
Group Project Assignment

Due Date: Monday Dec. 8, 2025, 11:59:59pm.

Additional Resources

- Kessler Game Github: https://github.com/ThalesGroup/kessler-game/tree/main
- XFC Youtube Channel: https://www.youtube.com/@fuzzycompetition8326
- Calculating intercepts in 2D space: https://www.codeproject.com/Articles/990452/Interception-of-Two-Moving-Objects-in-D-Space
- Another Kessler Game Example: https://github.com/xfuzzycomp/KesslerGameExample
- Kessler Game Development Manual by Dr. Dick (on Canvas)
- Dr. Dick's Kessler Game Agent (on Canvas)

Summary
You will work in teams of three to build a fuzzy control agent to play the Kessler implementation of the Asteroids arcade game, version 2.4.0, written by Thales North America as a part of the Explainable Fuzzy Challenge (XFC) competition at the NAFIPS annual conference. The agent must at a minimum determine the *thrust*, *turn_rate, fire* and *drop_mine* control outputs using a fuzzy controller; for full credit, the fuzzy system must be optimized by a genetic algorithm, and must be able to defeat the trivial agent test_controller.py from the Kessler GitHub examples directory. There will be an ***optional*** competition between student group agents; an off-line round-robin stage will be followed by a live single-elimination round of 16 on Friday Dec. 5, 5-9pm (pizza and soda provided).

Discussion
As discussed in class, this assignment is designed to exercise your skills in engineering a practical AI system as part of an existing project. The use case that you are implementing is the control agent for one player in the Asteroids arcade game (we will test your code in groups of two). We will use Thales North America's Kessler Game implementation of Asteroids, which is an entirely Python-based implementation from the ground up. Before continuing to read this assignment, we suggest you visit PyPi and download the current Kessler Game wheel file (pre-built executable). Install it using pip install as per the instructions on the site; more details are also available in the Kessler development manual written by Dr. Dick.

The /examples/ subdirectory of the Kessler Github contains the Python code needed to run a simple example of the game: a controller that just constantly spins in place, and shoots. The file defining this behaviour is test_controller.py; the vital method within it is actions(), which is called at every time step of the game. This file accepts the current game state and state of the player's ship, and needs to return four values: a floating point value defining how much thrust to apply on the ship's current heading in m/sec^2 (*thrust*); a floating point value indicating the turn rate to be applied in degrees per sec (*turn_rate*); a Boolean indicating whether or not to fire a bullet (*fire*); and a Boolean indicating whether or not to drop a mine at the current location. As you can see, the simple controller sets a *thrust* of zero (ship does not move); a *turn_rate* of 90 degrees per second; *fire* is always True; and *drop_mine* is always False. It is this method that you must overwrite with your own controller to complete this project. You will also need the scenario_test.py and graphics_both.py files to run the program. Download them to your Python development environment, and get the game running with the simple controllers as your first step. This simple controller is the agent you will need to defeat for full credit on the assignment.

Dr. Dick has completed a simple control agent of his own, that uses fuzzy logic to set a *turn_rate* and *fire* command designed to target and destroy the closest asteroid instead of constantly spinning. That agent is discussed at greater length in the Kessler Game Development Manual, posted on the Canvas page. This agent demonstrates many of the essential steps needed to complete your projects; however, as it does not move the ship (thrust is always zero), it would not itself be considered acceptable. Note that this agent also does not drop mines. Your agent will be required to control all four aspects of ship actions, meaning there must be rules that fire and yield non-zero actions for *thrust*, *turn_rate*, *fire* and *drop_mine* (although not necessarily all four at every moment).

There will be a competition held amongst any of the project teams that wish to join; any that do must submit the agent that will be used for competition by 11:59:59pm on Friday, Nov. 28, 2025. (You may still modify your agent for final submission, but we will not permit updates to the ***competition*** agents after this time.) Participating teams will be randomly divided into pools, and a head-to-head round-robin tournament will be conducted between pairs of agents by the teaching team. The agents will play five games in a head-to-head matchup with asteroids generated from test scenarios developed by Thales, with unlimited bullets and three lives; the winner will be the agent with the highest total score across all five games. Ties will be broken by the agent's average accuracy across all five games. This will narrow the field of competitors and produce seedings for a standard Round of 16 single-elimination tournament. The Round of 16 will take place Friday Dec. 5, from 5-9pm, (room TBD), with the competition following the same format as the round-robin, and being livestreamed to the room. Pizza and sodas will be provided. Members of the top three teams will receive extra credit points on their final grades:

1st place:	3 points
2nd place:	2 points
3rd place:	1 point

Requirements
Write a controller for the Kessler game in Python that at a minimum uses a declarative fuzzy controller to determine the four outputs of the actions() method: *trust*, *turn_rate*, *fire* and *drop_mine*. Your agent will be tested on the teaching team's own copy of the Kessler game, using different scenarios than what is provided in the examples directory. Higher marks will be achieved by using a genetic fuzzy tree to optimize your controller's performance; for full credit, your optimized agent will need to defeat the simple example agent under the same rules as the optional tournament: 5 game runs, with unlimited bullets and three lives, winners judged by total score across the five runs, ties broken by average accuracy.

Submission
Submit your code as a Python file (.py, NOT a Jupyter notebook) via the Canvas lecture page before 11:59:59pm on Dec. 8. If you wish to participate in the competition, submit the competition version of your agent to the separate "competition" submission link before 11:59:59pm on Nov. 28.

Grading

Agent uses a fuzzy controller and can play the game	75%
Agent optimizes a genetic fuzzy tree	15%
Agent defeats the agent from test_controller.py	10%