UNIVERSITY OF WOLVERHAMPTON
KNOWLEDGE ▪ INNOVATION ▪ ENTERPRISE

6CS014 – Complex Systems

Module Assessment

Module Code	6CS014.
Module Leader	Martin Jackson.
Semester	1.
Year	2020-21.
Assessment	A portfolio of work.
% of module mark	100%.
Set Date	Lecture 1.
Due Date	See the individual portfolio tasks.
Hand-in – what?	Written reports and application.
Hand-in- where?	Reports: Via Canvas. Software: Via Canvas.
Pass mark	40% or above is required to pass the module.
Method of retrieval	Further work during resit period.
Feedback	* Available within 4 weeks of the submission date. * Formative feedback is also available during workshop sessions.
Collection of marked work	Available within 4 weeks of the submission date.

Learning Outcomes

At the completion of the module, the student is expected to be able to:

1. Analyse and model complex systems.
2. Design and develop software for the control of complex systems.
3. Evaluate complex systems with respect to speed, accuracy and other factors applicable to the problems being focussed on.
4. To demonstrate an awareness of the issues involved in controlling emerging autonomous technologies, robots and other devices.

Introduction and Assignment Overview

Complexity science is the scientific study of complex systems. These are systems with many parts that interact to produce global behaviour. These systems cannot easily be explained in terms of interactions between the individual elements. Complex systems are dynamic, unpredictable and have many elements. They exhibit emergent behaviour (rather than prescriptive behaviour) that is not easy to predict.

Complex systems can be found almost everywhere, examples include:

- Ecosystems,
- Cities and Businesses,
- Crowd Simulation,
- Our Brains.
- The Internet (not individual computer system).

In this module, you will explore and model complex systems. You will learn how to simulate (e.g. program) complex systems and research the topic in terms of artificial intelligence and emerging autonomous technologies.

The module assignment involves a portfolio of work to be **individually** completed. The portfolio comprises of two tasks and provides overall evidence of your development in the area of complex systems. The portfolio should include an application and a research report. **Please see the assignment specification below for more details.** The purpose of the assignment is to allow students to explore automated agents, software simulation of complex systems and artificial intelligence topics.

><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><

Assignment Details

Portfolio Task 1: Design and Build an Automated Agent

Indicative portfolio task weighting: 60%.
Due date: 18/12/20 – Teaching week 12, Friday.
What to submit?:

An **individual** software development project. See below for more details. You are also required to submit a short video of your agents implementing their behaviour.

Word limit: N/A.

For this portfolio task you are required to design and build a simulation (i.e. a virtual environment) and an automated virtual agent (i.e. robot that uses the virtual environment) that will act and react to its environment. By the end of this task you will have created a software simulator that simulates a virtual environment. This task provides students with the practical experience of developing a software simulator and virtual agent that will demonstrate the student's knowledge and skills in the field.

You should build and document an autonomous agent written using the module tools, namely, Unity and C#. Below is a detailed overview of key elements of this portfolio task. This task is split into three parts. Each part requires an agent to be developed (i.e. coded).

Virtual Agent and Virtual Environment

In the future robots or drones may deliver or collect things (e.g. parcels or people). In this assignment you are required to develop an autonomous automated agent that represents a **delivery or collection agent**. In general, the agent should collect or deliver things from a start location. It should move to one or more delivery or collection locations. The agent should then return to the start location. The agent should act and react to its environment. **See the parts below for more detail.**

For simplicity, in this document I will refer to this scenario as a delivery or collection agent; however, you are encouraged to apply a theme of your choice to the simulation and virtual agent. Students are encouraged to be creative but should bear in mind the feasibility of the implementation and the requirements in this document as a constraint.

Here are some examples of possible agent themes. They are split into two types, delivery and collection. In a **delivery scenario** an item or items are transported to a goal(s) and the delivery agent returns to a home location. In a **collection scenario** an item or items are picked up from a goal(s) and the collection agent returns to a home location. They are essentially the same scenario; however, in the **delivery scenario** an agent has items before the goal(s) are reached. In the **collection scenario** the agent has items after the goal(s) have been reached.

Delivery scenario examples:

- Humanoid agent delivering parcels.
- Vehicle (e.g. lorry) agent delivering parcels.
- Animal storing food at locations for the winter.
- Drone agent delivering parcels.
- Simple taxi driver dropping off passenger(s).
    - No passenger pick-up required. Taxi already has passenger(s).

Collection scenario examples:

- Lumberjack cutting down trees and returning to a sawmill.
- Animal collecting food and returning to a home.
- Humanoid or vehicle (e.g. lorry) agent collecting parcels.
- Simple taxi driver collecting passenger(s).
    - No passenger drop-off required. Passengers are taken to taxi home location.

Your also need to create a simple virtual environment and choose an environment theme. Typically, this would match the agent theme. For example:

- Small area with house or drop-boxes / points.
- Larger area with roads, buildings or drop-boxes / points.
- Forest with trees and sawmill.
- Land animal environment (e.g. ants or tigers), home location and food locations.
- Underwater animal environment (e.g. fish), home location and food locations.
- Air environment (e.g. birds), home location and food locations.
- Medieval (e.g. Footman collecting food).
- Futuristic (e.g. robots).

You are encouraged to apply a theme to your agent and environment; however, a simple robot delivery theme is fine. **In addition, you must only implement <u>one</u> type of agent (e.g. delivery or collection) for all parts of the assignment. <u>You must keep the agent type and environment theme the same for all parts of the assignment.</u>**

You need to create a waypoint graph representation of the environment that your agents will use to navigate to locations. Your waypoint graph should have at least **10 nodes**. You should create a **<u>standard waypoint graph</u>** with **at least 10 nodes** and a reasonable set of connections (i.e. edges) between the nodes. The graph should be able to work with a standard pathfinding algorithm. A user should be able set a start location and multiple goal locations (**up to 5 goal locations**). A user should be able to do this via an interface or the Unity Inspector. The start location cannot be a goal location.

You should implement the world representation (waypoint graph / map) and pathfinding algorithm yourself. You should not use navigation methods built into Unity. If you do use built-in Unity techniques, you will receive 0% for the waypoint graph elements of the assignment.

Measuring Performance

The performance of all the agents you create should be measured during all their runs / trips / journeys. Performance measures should include time and total distance. During the simulation, a stopwatch should be used to time the duration for the total task time. The output from performance measures can be displayed via a user interface or printed to the Unity console

Part 1 - Virtual Agent for a Known Environment and Single Delivery or Collection Location

You are required to develop an autonomous automated agent that represents a delivery **or** collection agent (e.g. robot). The agent should start at a home location and deliver **or** collect an item (e.g. parcel) from a goal location.

The agent you create should be able to navigate a waypoint graph representation of your virtual environment. The agent should be able to move from a start location (e.g. a home location or any node on the graph) and

navigate (using a pathfinding technique) to another node on the graph (e.g. a delivery location / waypoint node). When the agent has reached their target, they should deliver or collect their item and return to the start location.

Once core functionality has been implemented you can extend the agents abilities. For example, navigate to more locations once the agent has returned to the start location or respond to event in environment (e.g. stop at traffic lights or unforeseen obstacles in its way) or multiple start locations.

The act of delivering or collecting an item does not need to be animated / virtually recreated. For example, in a delivery scenario you can simply use a box / parcel 3D model that disappears into the delivery location.

The agent in this part of the task has full knowledge of the virtual environment. The agent can see the whole environment and can use traditional navigation algorithms to move to target locations. The agent should be able navigate using **one** of the following techniques:

- Standard waypoint graph and A* pathfinding.
- Simple waypoint graph and straight-line movement.
    - *Note, you will only receive a pass mark for the waypoint graph elements of the assignment if you use this technique.*

Don't forget to include performance information (see above) in the agent code. **Please prefix your Unity scene name and main agent C# class name with Part1_.** For example, your scene could be called Part1_Town and your agent could be call Part1_Agent.

Part 2 - Collaborative Virtual Agent for a Known Environment and Single Delivery Location with Multiple Items

The second agent builds upon the work you completed for the first agent. You should use the same agent and environment theme. However, note in this scenario the agent has different abilities / behaviour. **Therefore, you should make a copy of your scene and agent from part 1 and adapt it. Or copy your scene and start your agent from again scratch using the knowledge you have gained from part 1.**

The agent in this part should have the same abilities as part 1; however, the agent should also be able to act and re-act to obstacles and other instances of itself in the environment.

In addition, the agent should be able to carry up to 10 items (e.g. parcels). The number of items an agent is going to carry should be able to be set in the Inspector or via a user interface. The speed of the agent should be affected by the number of parcels it is carrying. The speed of the agent should drop 10% for each parcel the agent is carrying. The speed of the agent should drop a maximum of 90% its original speed. The agent should also include some sort of visual representation of the items it is carrying. For example, you could add a 3D model for each parcel the agent is carrying, or you could simply output a number above the agent to indicate the number of parcels it holds.

The agent should monitor its environment for other agents (i.e. other instance of itself). The agent should collaborate with other agents (i.e. other instance of itself) by monitoring its separation and speed. The agent should adjust its speed (generally slow) when other agents come within a safety zone. If other agents get too close the agent should stop. The agents should communicate and collaborate / negotiate in order to allow one agent to keep moving. You don't want both agents to stop and continually block each other and the waypoint map forever. You should be able to add at least three instances of the same agent (i.e. game object / class) into the environment.

Your collaboration algorithm should consider the agents speed and immediate direction / destination when deciding which agent should move first / have right of way. Remember, your agents speed should slow based on the number of items it is carrying. Therefore, you want to avoid a slow agent blocking a faster agent.

Once agents have interacted and collaborated, they should adjust their outward behaviour to illustrate how the collaborated. For example, an agent could stop before a waypoint node to let another agent reach the node and travel down a different connection or the agents could rerun pathfinding to find alternative routes. Higher grades will be awarded for agents that have robust and sophisticated collaboration algorithms and outward behaviours.

Don't forget to include performance information (see above) in the agent code. You should also output information about collaboration encounters between agents. **Please prefix your Unity scene name and main agent C# class name with Part2_.** For example, your scene could be called Part2_Town and your agent could be call Part2_Agent.

Part 3 - Agent for a Known Environment and Multiple Delivery or Collection Locations

The third agent explores the scenario of multiple delivery locations. This agent makes use of techniques you completed for the first and second agents; however, it will have a different structure. You are required to develop an autonomous automated agent that represents a delivery **or** collection agent (e.g. robot) that has **multiple** delivery or collection locations. The agent should start at a home location and deliver **or** collect items (e.g. parcels) from multiple goal locations. The agent should be able to act and re-act to obstacles and other instances of itself in the environment.

The agent should be able to deliver or collect an item (e.g. parcel or food) from up to **5 goal locations**. This means you should be able to set up to **5 goal locations** (e.g. delivery or collection locations) in your waypoint graph.

The agent should plan the shortest route that starts at the start location (e.g. delivery depot), visits each delivery or drop-off location and returns to the start location.

This type of problem is time-consuming to solve using traditional pathfinding techniques. It would take a long time to calculate the overall shortest route. Therefore, the agent should use the following technique to solve this problem:

- Ant Colony Optimisation.

You should implement the Ant Colony Optimisation algorithm yourself. You should not use prebuilt tools to implement the algorithm. If you do use prebuilt, you will be capped at a pass for the Ant Colony Optimisation elements of the assignment.

You should still use A* pathfinding to determine routes / connections between goal (e.g. delivery) locations. **The agent should also have the collaborative abilities of the agent you developed for part 2.** You should be able to add at least three agents into the environment.

The user should be able to set the values for key properties in of the Ant Colony Optimisation algorithm in the inspector or via a user interface. For example, Q, alpha and beta.

Don't forget to include performance information (see above) in the agent code. You should also output information about collaboration encounters between agents. **Please prefix your Unity scene name and main agent C# class name with Part1_.** For example, your scene could be called Part1_Town and your agent could be call Part1_Agent.

For this portfolio task you are required to submit a screen capture video demonstrating all agent actions. The video should be no longer than 180 seconds. Please do not zip the video file when you submit it. You should use video capture packages; such as CamStudio (http://camstudio.org/), to record the video of your game prototype. If you have any questions or concerns about this, please speak to a member of the module team.

For this portfolio learning task, you may be required to demonstrate the agents you have developed to a member of the module team. This may happen for a number of reasons; for example, if you have not named your Unity scene name and main agent C# class clearly. If asked to demo you should come prepared to present a live demo of your agents and answer questions related to their development. For example, your applications and agents' structure, understanding of techniques and principles used.

What to Hand-in?

When you have completed the work, you should submit two files through the portal on Canvas. Please submit the following:

- A .zip file containing your Unity project.
- A screen capture video demonstrating all agent actions. The video should be no longer than 120 seconds. Please do not zip this file.

><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><

Portfolio Task 2: Investigation of Complex Systems with Respect to Artificial Intelligence and Emerging Autonomous Technologies

Indicative portfolio task weighting: 40%.
Due date: 18/12/20 – Teaching week 12, Friday.
What to submit?: An **individual** academic **report**.
Word limit: Approx 2,500 words.

The aim of portfolio task 2 is to learn about and demonstrate evidence of learning about complex systems with respect to artificial Intelligence and emerging autonomous technologies. You are required to research a complex systems topic and present your findings in the form of an academic report. Below are examples of possible report topics. **You can select your own title; however, you must get your title agreed by a module tutor.**

Examples of research report topics:

- Deep neural nets
- Autonomous flying vehicles
- Smart robots
- Autonomous space rovers
- Virtual assistants
- Etc.

You should produce a report of **approximately 2500 words** (about four pages of A4 using a 12pt. font, not including figures). The report should contain the following sections:

- **Introduction** – this should include an introduction to your chosen topic and explain why it is important. It should, in straightforward style state the purpose of the report / the problem being investigated.
- **Aim and Objectives** – include a clear section the outlines the aims and objectives of the report.
- **Literature Review** - An in-depth investigation of your chosen topic area, which draws upon a wide range of academic and research-based sources.
- **Analysis of Findings** – An analysis and evaluation of the research found during the literature review. This should be related to the aims of the report.
- **Conclusion** - A conclusion that draws out your findings.
- **References** - A list of references. Correct use of the Harvard referencing system.

What to Hand-in?

When you have completed the work, you should submit word file containing your report through the portal on Canvas.

><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><

General Important Information about the Assignment

Please read the key points below:

1. You must use the prescribed module tools to develop your game. These are: Unity, **version 2020.1.1f1 and C#**. If you do not use these tools you may fail the module.

2. You should use resources on the web to support your learning and development as a game's developer / programmer. However, you are **not permitted to copy online resources in part or entirety and submit them as your work**. This also includes following online tutorials in part or entirety and submitting them as your work. **The agents you create for this portfolio should be designed and developed specificity for this portfolio. You should start your development from an empty Unity project and create code based on your own knowledge and understanding. However, you should/can use pre-made graphical assets.** Please see the information at the end of this document and follow this link for more information regarding academic misconduct: http://www2.wlv.ac.uk/webteam/curr_sdts/sharpen/ss-HowtoAvoidAM.pdf.

><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><

Marking Schemes

The marking schemes / grids for each portfolio task can be found in the same folder as the assessment on Canvas.

Please read the marking grids carefully. Please speak to the module leader if you have any questions.

Remember – *Grades are not mechanically calculated but are a matter of academic judgement based on the marking criteria.*

You should be able to find the following marking grids for this portfolio:

- Marking Grid - Portfolio Learning Task 1: Simulator and Agent (Indicative weighting 60%).
- Marking Grid - Portfolio Learning Task 2: Investigation of Complex Systems (Indicative weighting 40%).

><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><