Modern Computer Games

COMP 521, Fall 2025

Assignment 4

Due date: Wednesday, November 26, 2025, by 11:59pm

Note: Late assignments will only be accepted with prior **written** permission of the instructor. Please make sure your code is in a professional style: **well-commented,** properly structured, and appropriate symbol names. Marks will be very generously deducted if not!

Description

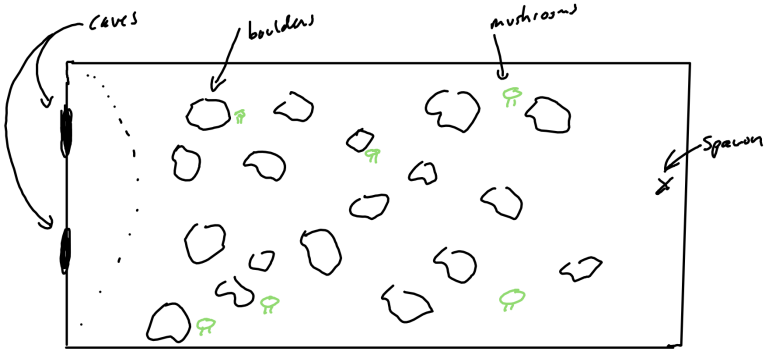
In this assignment you will build a goal-oriented AI for a game scene.
Note that you must implement the AI system entirely yourself: **do not use any built-in or external implementations, assets, or tools for game AI**. Note that this does **not** include pathfinding—you may use Unity's NavMesh or any other pathfinding implementation for navigation.

1. Build a basic game level, structured as a bounded, large rectangular outdoor area with two caves at one end. **4**

   Within this area add 10–20 obstacles representing boulders, large enough to hide the player from an ogre's view (see below). Obstacles should be randomly placed (different on every playthrough) but not overlap, and note that the area near the caves should be free of obstacles and that the cave area must be reachable from the player spawn-point.

   Some number (5–10) of mushrooms should also be spawned—these do not represent a pathing hazard, but should not overlap with boulders or other mushrooms.

   The cave interior does not need to be modelled.



2. Use a first-person camera to represent the player, spawned at the point indicated. The player should be able to move around within the terrain (use WASD for movement), but not move through rocks, or the ogres (see below). Ensure the player can look around (mouse controls camera direction), and scale the player movement so it takes around 6–10s for the player to navigate from the spawn point to the caves. **3**

   The player can turn invisible, but it is a limited resource that can last for a total of 10s. Invisibility should start in the off state, and the space-bar toggles invisibility on/off. There should be some obvious indication of whether the player is in invisible mode or not (but you do need to display the resource remaining). **3**

   A player has 2 lives, so it can withstand one attack from an ogre, but not two. Some obvious indication should be present of the number of lives remaining. **1**

   The player's goal is to steal treasure from the ogres (see below). Treasure is acquired by come close to (touching) a cave. Returning to the spawn-point with at least one treasure constitutes a win, and being killed by an ogre constitutes a loss. **1**

3. In each cave lives an ogre, each guarding their treasure just inside the cave. The ogres mostly hang around in front of their caves, but sometimes get hungry and go hunting for mushrooms. They also dislike the player, especially if the player takes their treasure.

Ogres should be controlled by an HTN-based AI (both ogres use the same HTN). Design an HTN that incorporates the following behaviours: **11**

- Ogres should have at last 2 distinct idle behaviours. Note that this assignment does not presume you use any kind of detailed animation, but you must ensure the idle behaviours are visually apparent.
- Ogres should be able to forage for mushrooms as a 3rd idle behaviour.
- Ogres should be able to find, pick-up, and throw a boulder at the player, as well as one other form of attack. (nb: To keep it simple, ogres themselves are not affected ogre attacks, including thrown boulders.)
- Ogres should have a limited field of view and a range of around 1/2 of the game level. Ogres should move close to the same speed as the player.
- Ogres are immediately aware if their treasure is taken, irrespective of their location, but cannot see an invisible player.

Your HTN design should incorporate multiple methods. In a *separate document* describe the HTN: the world state vector, the HTN structure itself, and indicate all pre/post-conditions.

Implement a simple, total-order forward planner for your HTN. **5**

The current plan being followed should be visually shown at all times: the ordered sequence of tasks to be done should be clear, as well as which step the ogre is performing. **2**

Note that these requirements necessitate some creativity in design. Treat it as a problem of building a scenario where the ogres can at least sometimes successfully defend their treasures from the player, but also where the player can usually, but not trivially, win. It should be possible for the player to observe all ogre behaviours (but perhaps in more than one play-through).

What to hand in

Assignments must be submitted on the due date **before midnight**. Submit your assignment to *MyCourses*. Note that clock accuracy varies, and late assignments will not be accepted without a medical note: **do not wait until the last minute**.

Include all source code necessary to run your simulation, as well as the document describing your AI design.

This assignment is worth 15% of your final grade. **30**

Programming assessment

	Mastery	Proficient	Developing	Beginning
Correctness	The solution works correctly on all inputs and meets all specifications.	The solution meets most of the specifications; minor errors exist.	The solution is incorrect in many instances.	The solution does not run or is mostly incorrect.
Readability	Well organized according to course expectations and easy to follow without additional context.	Mostly organized according to course expectations and easy to follow for someone with context.	Readable only by someone who knows what it is supposed to be doing.	Poorly organized and very difficult to read.
Algorithm Design	The choice of algorithms, data structures, or implementation techniques is very appropriate to the problem.	The choice of algorithms, data structures, or implementation techniques is mostly appropriate to the problem.	The choice of algorithms, data structures, or implementation techniques is mostly inappropriate to the problem.	Fails to present a coherent algorithm or solution.
Documentation	The solution is well documented according to course expectations.	The solution is well documented according to course expectations.	The solution documentation lacks relevancy or disagrees with course expectations.	The solution lacks documentation.
Performance	The solution meets all performance expectations	The solution meets most performance expectations	The solution meets few performance expectations	The solution is not performant.