Modern Computer Games

COMP 521, Fall 2020

Assignment 4

Due date: Monday, December 7, 2020, by 6:00pm

Note: Late assignments will only be accepted with prior **written** permission of the instructor. You must **explain** all answers and **show all work** to get full marks! 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

This assignment should be done in Unity as a 3D game level.

1. You will first need a basic game level. This assignment requires a bounded area containing several large rocks a few crates, and at least 5 mice (see below) running around the area. On one side of the area is a large (but not deep) cave/grotto (large enough for a player to enter but otherwise the interior is not complex). The player character (see below) spawns from an entrance on a side of the area adjacent to the cave entrance, as per the sketch below. Other than the player entrance and cave, all other level boundaries consist of high, impassable vertical obstructions. **3**

   Rock and crate distribution is randomized on each playthrough, but rocks and crates should never overlap. **3**

   There must be multiple routes between the player entrance and the cave, but it should not be possible for the player (see below) to walk directly from the player entrance to the cave. **4**

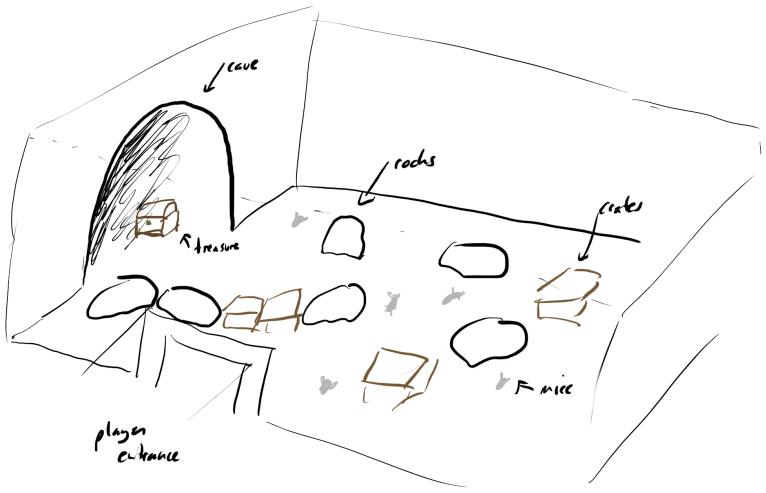

Figure 1: Sketch of level design. Not to scale; ensure there is adequate room for characters to move around.

2. The player's task is to take (by walking over) a treasure located in the cave and return to the level entrance. Spawn the player at the entrance. The player should be able to move through the terrain (use wasd for movement) but not pass through (or move) rocks, crates, or the cave monster (see below). Ensure the player can look around as well as move (ie the mouse controls camera direction). Scale the player movement so absent any monster activity (see below) it typically takes more than 10s to get from the entrance to the cave. **3**

   The player will need a defense value (initially 10) representing a shield resource—if the defense value is > 0, pressing the spacebar toggles activation of the shield, which then decreases the defense value at a rate of 1/s, turning off the shield if toggled or once the defense value reaches 0. There should be some visible representation of the shield being active and the defense value. **3**

   Once the player has moved over the treasure and returned to the entranceway, or been successfully hit twice (see below) the simulation should stop (game over). **1**

3. Guarding the area from the player is a large cave monster, emerging from the cave when the player enters the area. The monster is controlled by an HTN-based AI with the following possible behaviours, each of which is followed by a 1s cool-down before another action can tried. **4**

   - At least one idle behaviour that includes moving between locations.
   - At least two forms of distance attack, one of which must involve using the rocks as projectiles, and one of which must involve using the crates. All projectiles disappear if they hit the player; rocks that do not hit the player are reusable, but crates are destroyed by any use.

   Your monster AI should be based on a forward HTN planner, as described in class. Ensure that all necessary primitive actions are possible. **5**

   Design an HTN tree that enables all monster behaviours. Each behaviour should include at least one compound task having multiple methods. In a *separate document* describe the world state vector you use, indicating the meaning of each value and its possible types. Draw the HTN tree and indicate any pre/post conditions. **3**

   The current plan being followed should be visually shown at all times—the ordered sequence of plan steps should be clear, as well at which step the monster is currently executing. **3**

   Note that these requirements necessitate some creativity in design. Treat it as a problem of building a combat scenario, where the monster generally tries to prevent the player from acquiring the treasure, offering meaningful combat in that the player can usually win if they attempt to do so, but it is not trivial. It should be possible for the player to observe all behaviours (perhaps in more than one play-through).

4. Mice must be entirely controlled by *steering forces*. They should generally wander around the terrain, with occasional brief pauses. **4**

   Mice must not collide with rocks or crates, or enter the cave or player entrance. Mice should *avoid* collisions with each other, the player, and the cave monster. A mouse hit by a projectile or trodden on by the cave monster should be removed from the game-play. **4**

   Note: for the mouse movement, if you wish you may base (not copy verbatim) your implementation on an existing implementation of steering behaviours (only). If so, the source must be publically available, and you must document/credit the source—failure to do so will result in 0 for this question.

What to hand in

Assignments must be submitted on the due date **before 6pm**. 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.

This assignment is worth 15% of your final grade. **40**