Modern Computer Games
COMP 521, Fall 2025
Assignment 1

Due: Friday, September 12, 2025, by 23:59:59

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

This assignment requires you implement a basic "mini-game" in Unity3D. You will thus need to install and become familiar with the Unity game development environment from "`http://unity3d.com/`". This is a commercial product, but for everything you do in this course the free (personal or student) version is more than sufficient. **Please follow the instructions given on MyCourses with respect to Unity version and use of Unity assets.**

The tasks below focus on gaining a basic familiarity with Unity and an ability to create game-related software prototypes. Aesthetics are not a factor in grading—solid objects should be opaque rather than wire-frame, but you do not need to use external models or textures; creative use of basic assets and asset-combinations can be used to accomplish all objectives.

Also note that the Unity site has links to helpful forums and short tutorials on using Unity for different purposes, and those will be your main resource for dealing with the software.

Overall design: The assignment requires your construct a single game level. The main, static play area should consist of a large rectangular area, styled as an outdoor terrain, fully surrounded by impassible terrain (mountains or walls) to ensure the player does not leave the area. The overall game space should be divided into three logical areas, starting, cavity, and goal, which they must progress through one by one. The player must only be able to progress to each area in turn, and should not be able to be able to return to the previous area once they decidedly move to the next area.

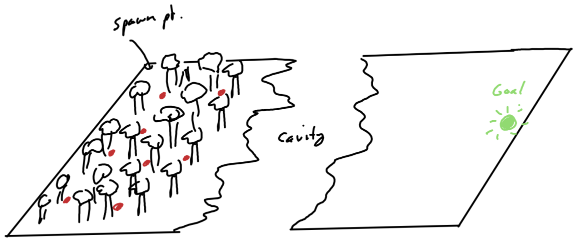

Figure 1: Level sketch; the player spawns in the top left, and must cross the cavity to get to the goal point.

Specific requirements

1. You must provide a non-trivial initial, static terrain as described above. Use a standard WASD/mouse controller for motion and looking around, with a first-person camera view (i.e., you do not need to code the camera and basic motion control, you can use the normal Unity assets). Left mouse click should be reserved for firing a projectile. Sufficient ambient or other light should be present so everything is visible.

   The game may terminate in a win or lose state; ensure that the result is apparent to the player.

2. **Starting area:** The player is spawned in one corner of a forested area. **7**

   Movement between the trees should be possible so the player can navigate to the objects and through the trees. As the player moves some evidence of where they were previously should also be (permanently) left behind, so it is clear to the player where they've been or not.

Scattered (randomly, different each time the game is run) are 10 objects the player can pick up by going over them. These objects should be visually obvious, relatively easy to find, disappear upon being picked up, and the count of objects the player has picked up should also be clear to the player.

The tree-line terminates at a large cavity/crevice/rift.

3. **Cavity area:** Outside the trees is a large, deep cavity, and it should be visibly obvious that the only way forward is to somehow cross it. However, moving into the cavity should cause the player to fall into it (game over). **9**

   The player here has to make a path across the cavity. They do so by "throwing" (i.e., shooting via the left mouse-button) the objects they picked up in the forest. As the thrown object would enter the cavity, it forms a platform the player can walk on, so a sequence of them can form a path across the cavity. Platforms should be 2D (or close to it) so overlap is not a concern.

   Objects thrown travel in a straight line away from the camera view at the time of the mouse click and move at a fast but discernible rate. If a thrown object encounters an obstacle (tree, ground, etc) or goes outside the level boundary (including a height threshold) it disappears. Only one object can be flight at a time, and each thrown object decrements the count of objects the player has.

   A path made from 5 platforms should be sufficient to cross the cavity. Platforms should be relatively proximal in order to form a walkable path, but need not be perfectly adjacent (tolerate some minor gaps). Individual platforms should only be usable once—once the player moves from platform A to platform B platform A should disappear, as an attempt to make it a one-way journey. Note that it should remain possible to walk off of a platform and fall into the cavity (game over).

4. **Goal area:** The other side of the cavity is the goal area, with an obvious destination at the far end. The player can freely move around this space, As the player moves toward the destination the cavity should expand into the goal area, effectively preventing backtracking. This growth can be coarse and discrete rather than continuous. Once the player reaches the goal the game should terminate (win state). **4**

What to hand in

You must submit your **well-commented and readable** Unity source code, along with a **separate document** providing a concise textual description of the state of your solution, describing any missing features or limitations.

For the Unity code itself, hand in a zip file of your project code containing all files needed in order to reconstruct and run your simulation. Please avoid excessive use of assets, especially high-resolution ones, as it easily results in very large archives. Allow ample time to upload your assignment solution, and please follow the general Unity instructions given in MyCourses.

For non-Unity questions, submit either an ASCII text document or a .pdf file *with all fonts embedded*. Do not submit .doc or .docx files. Images (plots or scans) are acceptable in all common graphic file formats.

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**.

This assignment is worth 5% of your final grade. **20**

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.