Description

This assignment requires you implement a relatively simple “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) 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 building game mechanics and structures. 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. Use a first-person perspective.

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 (i.e., include some low-height vegetation), fully surrounded by impassible terrain (mountains or walls) to ensure the player does not leave the area. The overall area is divided into two, disjoint areas (a starting area and a goal area) due to a deep and wide canyon; proportionally, the starting area occupies roughly about 50% of the level area, the chasm 30%, and the goal area 20%. Canyon walls may be either straight or more irregular.

![Level design (overhead view)](image1)

![Level design (profile)](image2)

Figure 1: Level design (overhead view)

Figure 2: Level design (profile)
The player should be spawned in one of the corners of the starting area, and has a goal of getting to the area across the chasm. The goal point should be visually obvious. The player should be able to move around freely in the starting area, at least within the first 1/2 of the starting area. As the player approaches the canyon, however, some number of trees should be dynamically instantiated, semi-blocking access to the canyon. This forested area should be a few trees deep, and must consist of at least \( n \) trees (see figures 1 and 2).

The player has the ability to fire projectiles, which can be used to destroy individual trees. Player projectiles should be visible objects moving rapidly (but noticeably) in a straight line out from the player in the direction of the camera. A projectile destroys the first tree it encounters. Only one projectile may exist at a time, and projectiles disappear upon hitting a tree, going out of bounds (including hitting the ground or going too high up), or if they go into the canyon area.

Once at the canyon the player should be able to observe a “bridge” across the canyon, but well below the cliff edge. The bridge itself is covered by short walls (high enough to prevent moving over, but low enough to see across) defining a grid-based perfect maze, 6 cells wide, and \( n \geq 10 \) cells long. The maze is constructed dynamically—each tree destroyed adds one complete row, starting from the player side. Thus upon getting through the forest the player should find just an empty bridge (no maze), or one with only a partially constructed maze, depending on how many trees they removed initially; once the maze is completed destroying further trees has no effect. Include walls on the sides of the bridge to prevent the player from falling off.

The player can walk off the cliff at any time. If they fall into the canyon itself, or onto an incomplete bridge-maze the game should end. If they fall onto the bridge with a completed maze they may begin to navigate the maze from wherever they land. Reaching any cell in the ending row of the maze should result in them teleporting up the goal area. If they jump/fall back into the canyon from the goal side, whether they land on the bridge or not, it is a game failure. Once they reach the goal location itself the game should end, with a conspicuous indication of success.

**Specific requirements**

1. You must provide a non-trivial initial, static terrain as described above. It should be bounded and styled as described above. Use the a standard WASD/mouse controller for motion and looking around, with a 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). Use a left mouse click for firing a projectile. Ensure sufficient ambient or other light is present so everything is visible.

2. Trees should be dynamically instantiated as the player approaches the canyon. Trees should be well placed, not significantly intersecting other trees (minor branch overlap is ok).

3. The player can fire one projectile at a time, which moves visibly and directly away from the camera view. Projectiles destroy the first tree they collide with, and otherwise disappear on contact with other level objects or if going out of bounds.

4. The maze structure is uniquely and randomly generated on each playthrough, with structure and behaviour as described above. You must implement your own maze generation code, but can use any appropriate algorithm. Specifically,
   
   (a) Maze walls should constrain movement (and prevent movement off of the bridge), but not occlude view. By looking down from the cliff edge it should be possible to see the entire maze (to the extent it has been constructed).

   (b) The \( n \) maze rows appear incrementally and dynamically as trees are destroyed. Partial maze constructions do not have to be solvable.

   (c) Once all rows have been created the maze should be a solvable, perfect \( 6 \times n \) maze with entrance and exit row on either side of the canyon. The choice of \( n \) is up to you and depends somewhat on canyon width and spacing, but ensure \( 10 \leq n \leq 15 \).

5. Winning and losing conditions should be properly detected and indicated.
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.

For the Unity questions, 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 high-resolution assets, 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.

This assignment is worth 15% of your final grade.