Modern Computer Games

COMP 521, Fall 2018

Assignment 1

Due date: Tuesday, Oct 2, 2018, 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 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 use the latest downloadable version (2018.2.8), and restrict your usage of Unity assets to free versions. Note that Unity is mainly supported on Windows and Mac, although Linux is possible as well.[1]

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 find or use external models or textures; creative use of basic assets and asset-combinations can be used to accomplish all objectives. Assume a first-person perspective is required unless otherwise specified.

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

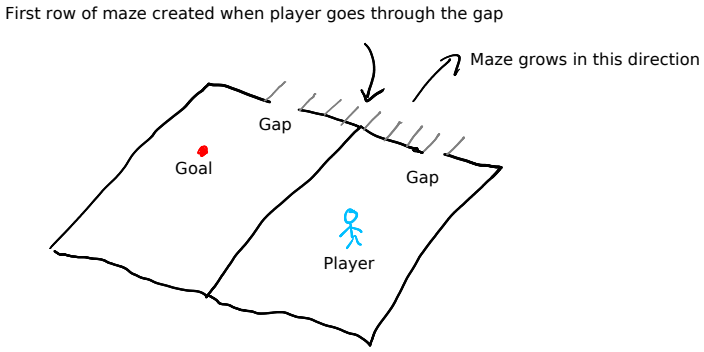
The main, static play area consists of a large area split into two (adjacent) rectangular terrains. Each is styled as an outdoor terrain (add some vegetation and ground texture), but is otherwise not overly occluded. The two terrains are separated by an unpassable barrier, which nevertheless allows the player to observe one terrain from the other. The other 3 sides of each terrain are surrounded by opaque unpassable barriers (such as walls or mountains), with the exception of a single gap in the barrier. The gaps face the same direction in each terrain, and each leads to a(n apparently) sheer drop (cliff edge).

The player starts in one of the regions, and should be able to see a conspicuous goal in the other region.

When the player first moves through the gap and over the cliff edge, instead of falling they enter a cell in the first row of an infinite maze. This maze is based on a grid structure, 8 cells wide against the cliff edge spanning/covering both gaps in the terrains (so the gaps are at opposite ends of the row). The maze has a floor (of course), and walls, but no ceiling, and no walls terminating the maze (ie you can still see past the maze and fall off the cliff by walking forward). Each additional row of the maze is constructed dynamically and incrementally as the player moves forward past the end of the current row (i.e., instead of falling a new row is constructed). Modulo use of the projectiles described below, the maze constructed should be randomly constructed so it is different in every play-through, (potentially/eventually) simple, and extend arbitrarily long. Use Eller's algorithm for generating the maze structure.

As well as moving around in the plane of the maze, pressing the spacebar should allow the player to jump vertically, going high enough that they can see the maze structure from above as they come down (they should be able to rotate, but not allowed to move laterally or jump while in the air).

[1] You will need to use version 2018.2.7f1. See thread `https://forum.unity.com/threads/unity-on-linux-release-notes-and-known-issues.350256`



Each cell of the maze is initially populated with a single projectile resource, shown as a small object. The player acquires these resources by moving over them (object disappears). The player can shoot a projectile (consuming a projectile resource) by pressing the "f" key (not while jumping). Fired projectiles should be clearly visible, move outward from the player in the lateral direction the player is facing at that point, and move about 2–3× the speed of the player.

Projectiles have two purposes. First, they can be used to short-circuit the maze traversal. Firing at least 3 projectiles at an inner maze wall (beyond the first row of maze cells) destroys that wall section. Second, firing a projectile out of the maze bounds, away from the terrain completes the maze, adding in terminal walls and ensuring the maze structure is simple (or would have been simple if maze walls were not destroyed). Projectiles fired at outer maze walls, or which go beyond the maze boundary in any other way just terminate.

Once the player gets through the maze and reaches the goal in the other region, the game terminates (win). A visual indication should be given, and no further action should be possible by the player.

Specific requirements

1. You must provide a non-trivial initial, static terrain as described above. It should be bounded and styled as described above. The gaps (maze entrance/exit) should be contiguous with the terrain. Use a standard WASD/mouse controller for motion and looking around, with a a first-person camera view. Ensure sufficient ambient light to easily see within the terrains and maze. **10**
2. The maze structure must be as described above. Note that the maze entrance/exit should be level and contiguous with the terrain(s) (it should be possible to see the terrain from the maze—construct the maze in place, do not teleport the player to a distinct space). **25**
3. Maze cells contain projectiles, which the player can pick up by passing over. The number of projectiles picked up bounds the number the projectiles that can be fired. **10**
4. Firing a projectile results in a visible projectile. Projectiles move laterally only, do not exceeed maze bounds, and react to maze walls as detailed above. **10**
5. Players can jump, reaching a height sufficient to observe the complete maze structure from above. **5**

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 an exported project containing all files needed in order to reconstruct and run your simulations. Note that Unity exports can be extremely large, and take non-trivial time to upload (another reason last-minute submission may not work out well).

This assignment is worth 15% of your final grade. **60**