Assignment 1 - Outdoor Environment, Stairs, Saving Level State, Object Animation
---
Due. Friday, February 12, 2021

1. Saving the State of a Game Level

The game is eventually going to involve multiple underground maze levels as were created for assignment 1 and an outdoor level. When the player leaves a level the state for each that level must be stored. This is so that the player can return to the level that they had previously visited. For this assignment there will only be two levels, the outdoor level in part 3 of the assignment, and one underground maze level.

You can choose to store the level information in memory or in a file.

All of the information needed to recreate the level must be stored. This includes the shape, location of the stairs, and any randomly placed items such as textures or blocks.

More features will be added to the levels in later assignments so you should write the storage code so that it can be extended to include more details. Things that are likely to be added include MOBS (moving monsters), treasure, and the location of stairs up and down.

2. Stairs

Create stairs that allow the player to move between levels. The player will start in the outdoor level. They should be able to move to the underground maze level and back up to the outdoor level when they stand on a staircase.

A down-staircase should be a gray cube. An up-staircase is a white cube. When the player steps on top of the staircase cube they should move to the next level up or down. This means that the state for the current level needs to be saved using the state-system developed in step 1 of this assignment. The level which the player is moving to should be either created (if the player had not previously visited it) or loaded using the state-system (if they had previously visited the level).

When a player moves up or down a level the viewpoint should be placed beside the staircase they just used.

More underground levels will be added to later assignments so the stair mechanism will need to work between more than the two levels in this assignment.

Randomly place the underground staircases in a room.

Do not simply put the outdoor level and the underground level in the world array. One level should exist at a time in the world[][][] array. When the player moves up or down a staircase then the state for the current level should be saved, the current level is then cleared from the game world, the level where the player is moving to should be either loaded or created in the world[][][] array, and then the player should be placed in the new level.

3. World Building

Create a three dimensional world from cubes. Use Perlin noise to
create an interesting surface for the world.

The upper surface of the world should be created using a Perlin noise function.
You can find an explanation of Perlin noise and code to implement it at:
https://en.wikipedia.org/wiki/Perlin_noise
You are not required to use the code from this site.

Pick a reasonable height for the terrain. It should be at least ten units tall.
Do not make the mountains too steep to climb. The player can only move up
one cube so a steep mountain will be impassible.

You don't need to make the mountains solid. You can create the outdoor
terrain so that it is one cube deep. You may find it useful to make the
terrain two or three units deep if the surface created using the Perlin noise
is particularly uneven.

Colour the surface so the lowest levels are brown, the middle levels are green,
and the tops of the hills are white.

Put a staircase on this level so the player can move to the underground
level. Start the player's viewpoint so they are near the staircase.

4. Clouds (object animation)

Clouds will be animated objects which move through the upper part
of the outdoor world space. You will need to keep track of the cloud's
locations and update the world array as they move. The movement of the clouds
should be easily visible. Pick speed and shape for clouds which clearly
demonstrate the motion. Make clouds larger than one or two cubes in size.

Add the clouds in the world array as white cubes.

Don't allow mountain tops to be destroyed by the clouds. Clouds should
pass through mountain tops if they collide with them. Alternatively, you
can just make sure the mountains and clouds never collide.

Note that timing control for animations that was discussed in class will be
included in a later assignment.

Starting Location

Start the viewpoint on the outdoor level, near the stairs down.

Choosing Parameters

It is important to pick values for parameters such as colours, speed of
objects, the effect of gravity so they are easy for the marker to see.
If the effect of a parameter it isn't obvious or is difficult to
see then it will be marked as missing or incomplete.

Make sure colours are distinct. Choose velocities for moving objects
that are fast enough to be seen.

Coding Practices

Write the code using standard stylistic practices. Use functions,
reasonable variable names, and consistent indentation.
If the code is difficult for the TA to understand then you
will lose marks.

As usual, keep backups of your work using source control software.

Starting Code

The starting code is available on the Courselink site.
You can untar the file using tar xf filename.
All of the changes to the code can be made in the a1.c file.

Note that the graphics code may be modified for later assignments. If you
make changes to the graphics code (in graphics.c or visible.c) then you
may have to recode the changes for a later assignment.

Submitting the Assignment

Put all of the files in a directory names 4820 so they unpack into
this directory.

Submit the assignment using Courselink. Submit only the source code,
the makefile, and any documentation. Bundle the code in a tar file.

Include a makefile that will compile the executable. Name the executable a1.
It's easier to always name the executable a1 so it wont change for
any of the assignments.

The TA will unpack your code and type "make". They will then try to
run an executable named "./a1". If the make command or executing a1
does not work then you will lose a substantial number of marks.

Don't name your program with a .exe extension. If the assignment
says name the executable a1 then don't change the name to a1.exe.

It is always a good idea to unpack and test the file you are submitting
to be sure that what you submit actually compiles.