COSC 435 Final Project Proposal
Mary Festa, Noah Hoff, Alexandra Nie, Bec Mitchell

Darwin Thinking Path:
Oh Deer! Natural Selection

Statement

Our goal is to design a maze game set on the Colgate nature trails up by the ski hill. The objective is to escape the maze before the timer runs out, which will be complicated by obstacles that the player meets along the way, such as geese, alcohol, and packs of Colgate students. The player will also be able to track their location on a map which updates as they traverse the map to show the locations they have already been, and collect power-ups such as coffee to increase speed, and a clock to extend the timer. The player will navigate through the maze, with the camera angle following the character to show a back view.

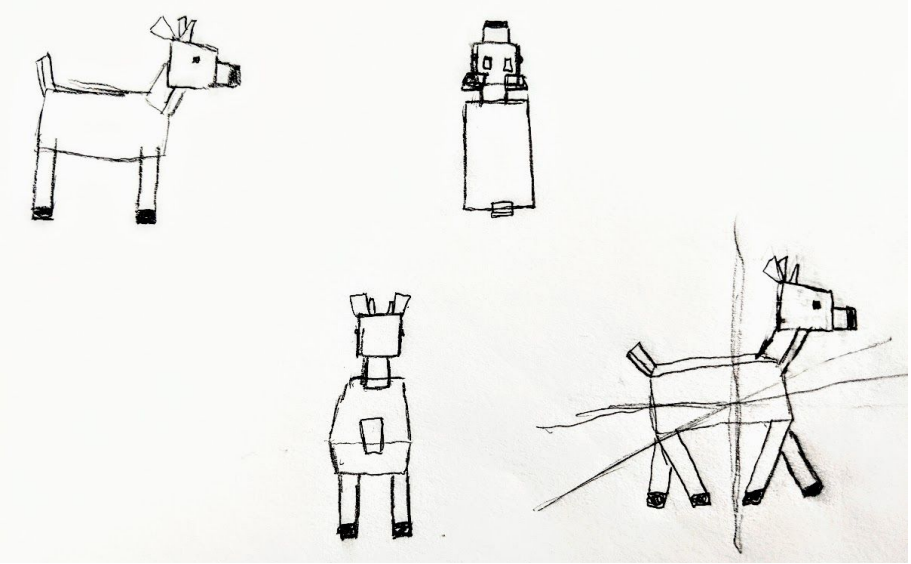
We will have different levels of difficulty, each with a theme corresponding to one of the either autumn or winter. The mazes will increase in size based on the level, and the number of obstacles and power ups will also increase.

We want to make sure that the gameplay is extremely clean, and prefer low-poly renderings of the objects in our scene. Less complexity in individual objects will allow us to focus on creating a game that runs smoothly and looks finished. In order to facilitate the creation of multiple levels, we plan on implementing a common practice of using an ASCII level editor to convert a grid of characters into our maze.

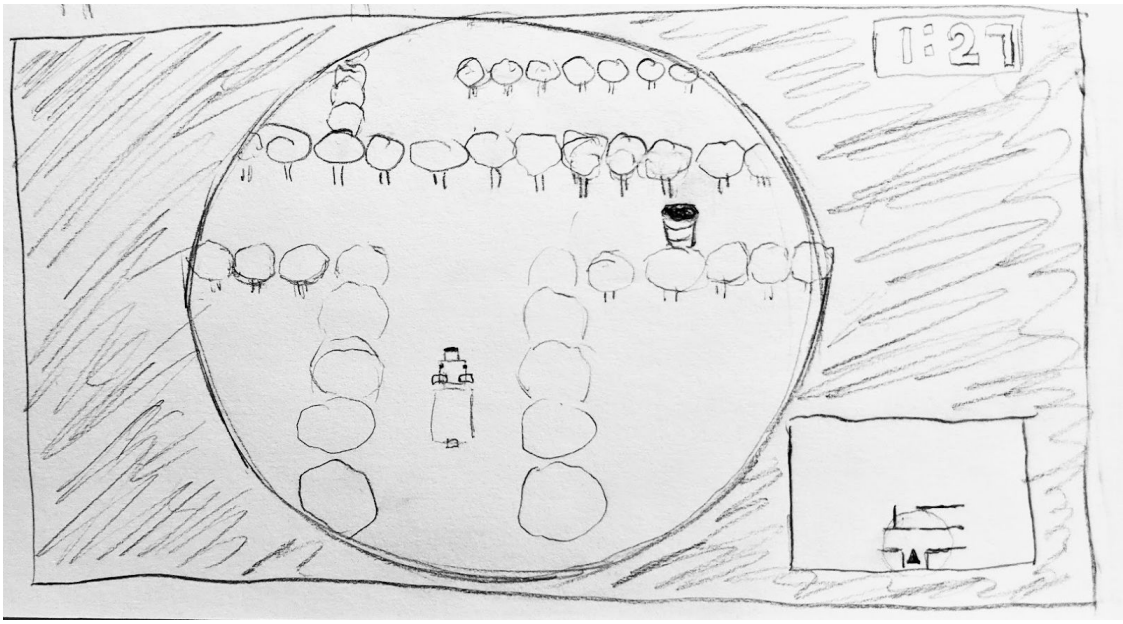
Technical Outline

1. **Background:** We will use two different backgrounds for different levels, representing autumn and winter at the nature trails. The background in this case would be the floor of the maze.

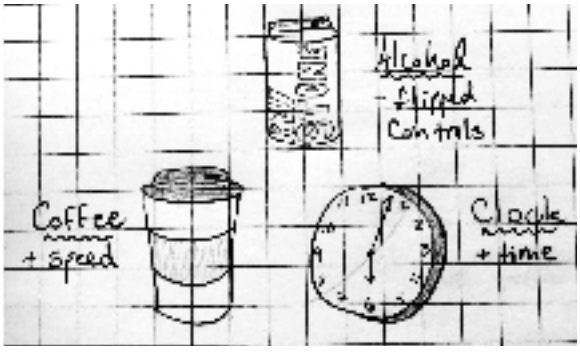
2. **Character:** The character will be modeled after a deer, and made up of basic 3D shapes for a block-style animation.



3. **Character movement:** The character will be able to move with the left, right, up, and down arrows to navigate the maze. We will animate the character so that its legs move appropriately with footsteps.
4. **Maze:** To create the maze, we plan on using a level editor that translates text into items on the map such as the one described at https://eloquentjavascript.net/16_game.html. This will facilitate the creation of the maze, and allow us to edit it more easily. We also plan to create a map which will update as the player traverses the maze, so that they can keep track of where they have been.

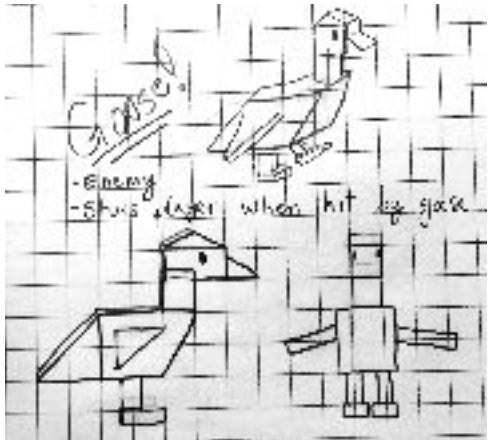


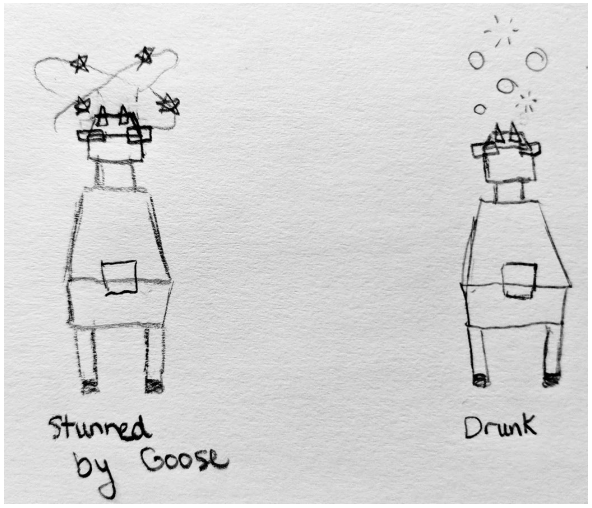
5. **Levels:** There will be multiple levels, representing different seasons (namely autumn and winter). Some models will remain the same between levels (e.g. your character), and others will change (e.g. trees becoming snowy/autumnal).
6. **Timer:** The game will be timed, and if you do not reach the finish point by the end of the timer you lose the game. We can use a loop function that will continue until the end of the time that we allotted is reached.
7. **Power-ups:** We will have power-ups that will increase your chances of reaching the end of the maze by the time allotted. These will give you the ability to increase your speed (coffee) or add more time to the clock (clock icon). This can be implemented by changing the speed that we had initially set the character to walk to a faster rate for a certain amount of time, and adding extra time to the timer that we had counting down, respectively.



8. **Obstacles:** We will have obstacles that will decrease your chances of reaching the end of the maze on time. These can cause actions such as being stunned from running into a goose (we will pause the character's movement for a certain length of time, without pausing the timer), being relocated randomly to a new place on the map from running into Colgate students, and being "drunk" from alcohol that was left (where we reverse the

key controls from the initial configuration for a set period of time).





Objectives

1. Model Colgate forest path setting by creating simple tree models using three.js with lambert shading, and repeating it to form forests. We can use low-poly trees modeled after shapes in the Shapes.html file (e.g. the icosahedron) that we worked with to achieve this. We can model the environment with variability (for example: the tree size, color, amount).
2. Model the main character (hero), which is a deer, using basic shapes (such as cubes and pyramids) created three.js with lambert shading.
3. Implement varying levels, each with a unique color palette depending on the season (winter or fall).
4. Implement particle effects in the form of snow and leaves, representing winter and fall, respectively.
5. Create appropriate 2D collisions with the walls of the maze, enemies, and power-ups. The character should be able to run up to walls but not move any further than that (and update

the top view of the map as it goes), it can walk on the ground, it can interact with enemies and power-ups (the game will react with the appropriate changes in effects on the hero or other gameplay effects).

6. Implement power-ups and obstacles that affect the player character. The two power ups would be coffee and a clock, both items found around the maze that would increase player speed and add time to the timer respectively. Obstacles would be another pickup, in the form of alcohol which would remap the players controls for a short period so they are backwards, and also enemies in the form of geese (which stun the player when they collide with them) and students (which relocate the deer to a random area of the map). These powerups and obstacles will be placed randomly, and will appear in different places of the maze each game.
7. Create fluid animations for the player character and the enemies in the maze. The enemies and player character would both have walking animations, but the player character would also have animations for completing the maze and failing to complete it in time. There will also be appropriate animations associated with the character when affected by an obstacle, such as bubbles for being "drunk," and stars above their head for being stunned by a goose.
8. Implement clean and clear UI elements which show the time left, what level the player is on, and a dynamic map which reveals more as the player explores.
9. Implement an immersive environment with lighting (with lambert shading) and shadows, using different light placements depending on the season and where the sun is in the scene. The GUI controls will be able to turn on and off the particle effects.
10. Generate a maze according to existing maze-algorithms such as the hunt and kill algorithm, and use a text-based level editor to convert ASCII characters in a grid to the grid representing the maze, adjusting the level of difficulty for each maze in different levels of the game.

References

1. Physijs: https://chandlerprall.github.io/Physijs/
2. 3D Physics Game: https://gamedevelopment.tutsplus.com/tutorials/creating-a-simple-3d-physics-game-using-threejs-and-physijs--cms-29453
3. Level editor:
    a. https://eloquentjavascript.net/16_game.html
    b. https://github.com/misha-codes/eloquent-javascript--level-editor
4. Snow and rain particle effects: https://stemkoski.github.io/Three.js/Particle-Engine.html
5. WebGL maze with map: http://www.madore.org/~david/misc/webglmaze.html

6. Simple animation tutorial:
   https://blog.openbloc.fr/a-simple-and-flexible-render-loop-in-three-js/
7. 3D trees:
   https://free3d.com/3d-model/low-poly-tree-449895.html
8. Three.js Primitives:
   https://threejsfundamentals.org/threejs/lessons/threejs-primitives.html
9. Low Poly Coffee Cup:
   https://www.vectorstock.com/royalty-free-vector/low-poly-isometric-coffee-cup-vector-15376486
10. Low Poly Bottle:
    https://yellowimages.com/images-360/products/low-poly-beer-bottle-yi3602240?ca=2_15
11. Hunt and kill maze algorithm
    https://weblog.jamisbuck.org/2011/1/24/maze-generation-hunt-and-kill-algorithm